CLUSTERING DOKUMEN SKRIPSI DENGAN MENGGUNAKAN *HIERARCHICAL AGGLOMERATIVE CLUSTERING*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Danang Aditya Wicaksana
NIM: 145150200111043

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

CLUSTERING DOKUMEN SKRIPSI DENGAN MENGGUNAKAN *HIERARCHICAL AGGLOMERATIVE CLUSTERING*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Danang Aditya Wicaksana
NIM: 145150200111043

Skripsi ini telah diuji dan dinyatakan lulus pada
23 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Putra Pandu Adikara, S.Kom, M.Kom	Sigit Adinugroho, S.Kom, M.Sc
NIP: 19850725 200812 1 002	NIK: 2016078807011001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP: 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 10 Juli 2018

Danang Aditya Wicaksana

NIM: 145150200111043

KATA PENGANTAR

Puji dan syukur ke hadirat Tuhan Yang Maha Esa atas segala rahmat yang telah diberikan-Nya sehingga skripsi yang berjudul "*Clustering* Dokumen Skripsi Dengan Menggunakan *Hierarchical Agglomerative Clustering*" ini dapat penulis selesaikan tepat waktu.

Tujuan dari penulisan skripsi ini adalah sebagai salah satu syarat untuk memperoleh gelar sarjana komputer dari Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Pengerjaan skripsi ini tentunya telah melibatkan banyak pihak yang sangat membantu dalam berbagai hal. Oleh karena itu, penulis ingin mengucapkan rasa terimakasih yang sebesar-besarnya kepada:

1. Bapak Putra Pandu Adikara, S.Kom, M.Kom dan Bapak Sigit Adinugroho, S.Kom, M.Sc selaku Pembimbing skripsi yang telah membimbing penulis dengan sabra dalam menyusun skripsi hingga selesai.
2. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku Ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang.
3. Bapak Tri Astoto Kurniawan, S.T, M.T, Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya Malang yang telah menyetujui permohonan penyusunan skripsi.
4. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D selaku Dekan Fakultas Ilmu Komputer Universitas Brawijaya Malang yang telah memberikan ijin penelitian.
5. Bapak Eko Subijantoro dan Ibu Endang Purwaningsih selaku orang tua penulis tercinta yang senantiasa memberikan doa dan dukungan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini tepat waktu.
6. Sendi, Iqbal, Adit, Josua, Bambang, Adhy dan Kamal selaku teman terdekat penulis selama di Malang yang bersama-sama menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya Malang.
7. Eka, Nisa, Ninda, Sarah, Isradi, Elha dan teman-teman kelas Informatika C tahun 2014 yang lain yang sudah membantu penulis beradaptasi dengan kehidupan di Kota Malang.
8. Rekan-rekan BIOS Exalt yang sudah memberikan pengalaman-pengalaman yang indah dan berharga selama satu periode.
9. Semua pihak yang banyak memberi bantuan yang tidak bisa disebutkan satu persatu.

Penulis menyadari bahwa penyusunan naskah skripsi ini masih jauh dari kata sempurna, sehingga saran dan kritik yang membangun sangat penulis harapkan.

Akhir kata, semoga skripsi ini dapat bermanfaat bagi semua pihak yang menggunakannya.

Malang, 10 Juli 2018

Penulis

danangadit26@gmail.com

ABSTRAK

Danang Aditya Wicaksana, *Clustering* Dokumen Skripsi Dengan Menggunakan *Hierarchical Agglomerative Clustering*

Pembimbing: Putra Pandu Adikara, S.Kom, M.Kom dan Sigit Adinugroho, S.Kom, M.Sc

Skripsi adalah suatu dokumen dari karya ilmiah yang disusun oleh mahasiswa pada tingkat strata 1 yang membahas suatu topik atau bidang tertentu dari hasil penelitian atau pengembangan yang telah dilakukan oleh mahasiswa tersebut guna mengikuti ujian akhir untuk memperoleh gelar sarjana. Pada Ruang Baca Fakultas Ilmu Komputer dan Perpusatakaan Pusat Universitas Brawijaya terdapat masalah yang timbul yaitu tidak ada pengkategorian seluruh dokumen skripsi yang disimpan. Metode *Hierarchical Agglomerative Clustering (HAC)* diimplementasikan untuk *clustering* dokumen skripsi berdasarkan judul skripsi. *HAC* mengelompokkan dokumen secara iterative mulai dari *cluster* terkecil hingga 1 *cluster* terbesar. *Input* data yaitu berupa judul dokumen skripsi Teknik Informatika Universitas Brawijaya. Tahap *preprocessing* dilakukan terhadap data judul skripsi untuk mendapatkan fitur berupa *term.* Seluruh *term* yang didapatkan diproses untuk mendapatkan bobot *TF-IDF*. Nilai kemiripan antar dokumen diperoleh dari nilai *cosine distance.* Proses *clustering* menggunakan 3 pilihan jarak sebagai parameter yaitu *single linkage, complete linkage* dan *average linkage*. Hasil *clustering* dari masing-masing parameter jarak ditampilkan label tiap *cluster* yang dihasilkan dan tiap *cluster* yang dihasilkan dievaluasi menggunakan *silhouette coefficient*. Dari hasil pengujian terhadap 100 dokumen skripsi diperoleh nilai *Silhouette Coefficient* dari *single linkage* adalah 0,10125*, complete linkage* adalah 0,155733 dan *average linkage* adalah 0,160428*. Average linkage* lebih baik dalam mengelompokkan dokumen dibandingkan *single linkage* dan *complete linkage.*

Kata kunci: *skripsi*, *hierarchical agglomerative clustering*, *silhouette coefficient*

ABSTRACT

Danang Aditya Wicaksana, *Cluster*ing Minor Thesis Document Using Hierarchical Agglomerative *Cluster*ing

Supervisor: Putra Pandu Adikara, S. Kom, M. Kom and Sigit Adinugroho, S. Kom, M.Sc

A minor thesis is a document of a scientific work compiled by a student at the level of stratum 1 which discusses a particular topic or field of research or development results that the student has undertaken in order to take the final examination to obtain a degree. In the Reading Room of the Faculty of Computer Science and the Central Library of Brawijaya University there is a problem that arises that there is no categorization of all minor thesis documents stored. Hierarchical Agglomerative Clustering (HAC) method is implemented for clustering minor thesis documents based on minor thesis title. HAC classifies iterative documents from the smallest cluster to the largest 1 cluster. Input data that is in the form of title of minor thesis document of Informatics Engineering Brawijaya University. The preprocessing stage is performed on the minor thesis title data to get the term feature. All the terms obtained are processed to get the weight of TF-IDF. The value of similarity between documents obtained from the value of cosine distance. The clustering process uses 3 distance options as the single linkage, complete linkage and average linkage parameters. The clustering results of each distance parameter are displayed on the label of each cluster generated and each cluster generated is evaluated using silhouette coefficient. From the test result on 100 minor thesis documents obtained the value of Silhouette Coefficient from single linkage is 0,10125, complete linkage is 0,155733 and average linkage is 0,160428. Average linkage is better in grouping documents than single linkage and complete linkage.

Keywords: *minor thesis, hierarchical agglomerative clustering, silhouette coefficient*.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

1.1 Latar belakang

Skripsi adalah suatu dokumen dari karya ilmiah yang disusun oleh mahasiswa pada tingkat strata 1 yang membahas suatu topik atau bidang tertentu dari hasil penelitian atau pengembangan yang telah dilakukan oleh mahasiswa tersebut guna mengikuti ujian akhir untuk memperoleh gelar sarjana (Huda, 2011). Setiap mahasiswa untuk memperoleh gelar sarjana dibutuhkan suatu dokumen skripsi. Sehingga skripsi merupakan suatu kewajiban yang harus dikerjakan untuk setiap mahasiswa strata 1. Jumlah mahasiswa setiap tahunnya pasti akan selalu bertambah dan mahasiswa yang lulus juga selalu bertambah. Bagi mereka yang sudah lulus berarti sudah menyelesaikan skripsinya. Dengan hal ini tentunya jumlah dokumen skripsi untuk setiap periode akan selalu bertambah. Dalam studi kasus di Ruang Baca Fakultas Ilmu Komputer Universitas Brawijaya maupun di Perpustakaan Pusat Universitas Brawijaya seluruh dokumen skripsi selain terdapat pengarsipan dokumen skripsi secara *hardcopy* juga terdapat pengarsipan *softcopy.* Ruang Baca Fakultas Ilmu Komputer Universitas Brawijaya dan Perpustakaan Pusat Universitas Brawijaya memliki *website* masing-masing untuk menampilkan dokumen skripsi yang tersedia. *Website* Ruang Baca Fakultas Ilmu Komputer dapat diakses pada alamat http://10,34.15.7/ namun hanya dapat diakses di lingkungan Universitas Brawijaya. Pada *website* tersebut tidak menyediakan seluruh isi dari dokumen skripsi namun hanya judul skripsi mahasiswa dan lingkup dokumen skripsi yang ditampilkan hanya dokumen-dokumen skripsi dari mahasiswa Fakultas Ilmu Komputer Universitas Brawijaya. *Website* Perpustakaan Pusat Universitas Brawijaya yang menampilkan kumpulan dokumen skripsi dapat diakses pada alamat http://digilib.ub.ac.id:81/bkg/. Pada *website* tersebut menampilkan seluruh dokumen skripsi dari seluruh mahasiswa Universitas Brawijaya. Informasi skripsi yang ditampilkan pada *website* tersebut hanya judul dan abstrak, tidak mencakup seluruh isi dari dokumen skripsi. *Website-website* tersebut memiliki kekurangan yaitu tidak terdapat pengkategorian kelompok dari dokumen-dokumen skripsi yang ada.

Bagi mahasiswa yang ada pada tingkat akhir memiliki kewajiban mengerjakan skripsi. Sebelum mengerjakan skripsi mereka diharuskan untuk mencari rujukan terhadap penelitian yang pernah dilakukan sebelumnya. Dalam mencari rujukan tersebut mahasiswa memerlukan beberapa referensi. Pada saat mencari referensi tersebut mahasiswa pada umumnya mengunjungi Perpustakaan maupun Ruang Baca. Namun bagi mahasiswa yang membutuhkan gambaran topik skripsi apa saja yang tersedia sedikit kesulitan dikarenakan di Ruang Baca Fakultas Ilmu Komputer Universitas Brawijaya maupun di Perpustakaan Pusat Universitas Brawijaya tidak terdapat kategori-kategori atau kelompok-kelompok dalam pengarsipan dokumen skripsi dalam bentuk *hardcopy* ataupun dalam informasi di *website* yang tersedia.

Tujuan utama dari penelitian ini yaitu mengelompokkan dokumen pada kategori yang tepat dan memunculkan label kategori dari kelompok-kelompok

seluruh dokumen skripsi mahasiswa Teknik Informatika Universitas Brawijaya. Metode pengelompokan dokumen skripsi yang digunakan adalah metode *clustering*. Metode *clustering* dipilih karena proses pengelompokan tidak memerlukan label kelas terlebih dahulu sehingga dokumen skripsi dapat secara otomatis dikelompokkan pada *cluster* yang memiliki kemiripan terdekat (Saad, Mohamed, & Al-Qutaish, 2012). Label kelas nantinya akan dimunculkan setelah dokumen-dokumen sudah terkelompokkan.

Metode *clustering* yang dipilih adalah *hierarchical agglomerative clustering*. Metode tersebut dipilih karena dapat mengelompokkan dokumen secara bertahap yaitu mulai dari klaster yang kecil atau klaster dengan jumlah dokumen yang masih sedikit bertahap hingga keseluruhan data yang ada terkelompokan menjadi 1 klaster besar (Han, Kamber, & Pei, 2012). Keuntungan dari penelitian tersebut yaitu peneliti tidak memerlukan inisialisasi jumlah *cluster* terlebih dahulu sehingga memungkinkan dokumen terkelompokan pada klaster yang tepat. Manfaat lain yang didapat selain dapat mengetahui kelompok-kelompok klaster yang dihasilkan yaitu dapat mengetahui nilai ketepatan suatu dokumen berada pada suatu klaster.

Terdapat penelitian lain yaitu *Hierarchical Document Clustering based on Cosine Similarity measure* pada tahun 2017 menjelaskan mengenai penerapan metode *Hierarchical Clustering* berdasarkan *Cosine Similarity measure* untuk *clustering* dokumen (Popat, Deshmukh, & Metre, 2017). Dalam penelitian tersebut membandingkan beberapa metode yang digunakan untuk *clustering* dokumen yaitu *Hierarchical Agglomerative Clustering (HAC), K-means, centroid similarity.* Nilai *accuracy* pada jumlah klaster 5 untuk metode *HAC* yaitu 0,872, metode *k-means* yaitu 0,708, metode *centroid similarity* yaitu 0,839. Nilai akurasi dari *HAC* selalu tertinggi setiap pertambahan jumlah klaster hingga jumlah klaster 10 dibandingkan dengan metode *K-means* dan *centroid similarity*. Saat jumlah klaster 10 nilai akurasi dari metode *HAC* yaitu 0,77, metode *k-means* yaitu 0,578 dan metode *centroid similarity* yaitu 0,701.

Penelitian lain tentang perbandingan metode *clustering* menggunakan metode *single linkage* dan *k-means* pada pengelompokan dokumen (Handoyo, Mangkudjaja, & Nasution, 2014). *Single linkage* merupakan metode pengambilan jarak terdekat dalam penerapan *Hierarchical Agglomerative Clustering.* Penelitian tersebut menggunakan metode pengujian *shilouette coefficient* dan *purity*. Dari hasil pengujian ketika dokumen berjumlah 100 nilai *silhouette coefficient* untuk metode *single linkage* pada klaster berjumlah 3 adalah 0,04 sedangkan untuk metode *k-means* pada jumlah klaster 3 bernilai 0,01. Nilai *silhouette coefficient* dari metode *single linkage* semakin bertambah ketika jumlah klaster semakin banyak sedangkan untuk metode *k-means* selalu tidak jauh dari nilai 0, Pada saat jumlah klaster 30, nilai *silhouette coefficient* dari *single linkage* adalah 0,24 sedangkan untuk *k-means* bernilai 0. Untuk nilai *purity* dari *single linkage* selalu 1 untuk jumlah klaster mulai dari 3 sampai dengan 30, Sedangkan untuk *k-means* nilai *purity* ketika klaster berjumlah 3 adalah 0,4 dan terus menurun ketika jumlah klaster terus bertambah.

Dari penelitian-penelitian tersebut dapat disimpulkan bahwa *Hierarchical Agglomerative Clustering* lebih unggul dari segi hasil analisis. Pada penelitian pertama menunjukkan bahwa *Hierarchical Agglomerative Clustering* lebih unggul dibandingkan dua metode *clustering* lain yaitu *K-means* dan *centroid similarity.* Pada penelitian kedua *Hierarchical Agglomerative Clustering* lebih unggul menurut nilai *silhouette coefficient* dan *purity*-nya dibandingkan dengan metode *k-means* dengan jumlah data 100 dokumen. Dari hasil yang disimpulkan dalam penelitian tersebut maka dipilihlah metode *Hierarchical Agglomerative Clustering* sebagai teknik untuk menyelesaikan permasalahan dalam penelitian *clustering* dokumen skripsi karena cukup relevan apabila dilihat dari jenis data yang digunakan yaitu berupa dokumen dan jumlah dokumen yang digunakan yaitu minimal 100 dokumen.

1.2 Rumusan masalah

Dari latar belakang yang sudah diuraikan diatas, didapatkan rumusan masalah sebagai berikut:

1. Bagaimana hasil *cluster* yang didapatkan dari *clustering* dokumen skripsi menggunakan *Hierarchical Agglomerative Clustering* berdasarkan parameter pengujian yang dilakukan?
2. Bagaimana ketepatan suatu dokumen dalam *cluster* berdasarkan parameter terbaik yang diujikan pada *clustering* dokumen skripsi menggunakan *Hierarchical Agglomerative Clustering*?
3. Bagaimana label *cluster* yang dihasilkan dari *clustering* dokumen skripsi menggunakan *Hierarchical Agglomerative Clustering* berdasarkan parameter pengujian terbaik?

1.3 Tujuan

Berdasarkan rumusan masalah yang telah dijabarkan, maka dapat dirumuskan tujuan penelitian sebagai berikut:

1. Mendapatkan parameter terbaik dari parameter-parameter yang diujikan pada *clustering* dokumen skripsi menggunakan *Hierarchical Agglomerative Clustering* dan mendapatkan hasil *cluster* terbaik berdasarkan evaluasi yang dilakukan pada setiap parameter.
2. Mengetahui nilai ketepatan suatu dokumen yang dikelompokkan pada suatu *cluster*.
3. Mendapatkan label tiap-tiap *cluster* dari penerapan metode *Hierarchical Agglomerative Clustering* dalam permasalahan *clustering* dokumen skripsi.

1.4 Manfaat

Manfaat dari penelitian ini adalah dokumen skripsi mahasiswa Teknik Informatika Universitas Brawijaya dapat terkelompokkan pada kategori yang tepat dan memunculkan label-label topik dokumen skripsi sehingga kelompok dokumen skripsi Teknik Informatika Universitas Brawijaya dapat dikenali. Manfaat lain

adalah untuk mengetahui apakah metode *Hierarchical Agglomerative Clustering* dapat menghasilkan kelompok-kelompok topik dokumen skripsi dengan tepat.

1.5 Batasan masalah

Agar penelitian memiliki fokus yang jelas, maka dirumuskan batasan masalah sebagai berikut:

1. Dokumen skripsi yang digunakan sebanyak 100 dokumen.
2. Dokumen skripsi yang digunakan berasal dari Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
3. Data yang diambil berupa judul skripsi.
4. Data diperoleh dari *website* Perpustakaan Pusat Universitas Brawijaya http://digilib.ub.ac.id:81/bkg/.
5. Data diambil dengan kata kunci "Teknik informatika" dan diambil judulnya secara acak.

1.6 Sistematika pembahasan

Sistematika penulisan penelitian ini secara garis besar akan diuraikan sebagai berikut:

BAB 1 PENDAHULUAN

Berisi pembahasan tentang latar belakang dilakukannya penelitian ini, rumusan masalah yang akan dibahas, tujuan dan manfaat yang akan dicapai dari penelitian ini, Batasan masalah dalam penelitian dan sistematika dari *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering*.

BAB 2 LANDASAN KEPUSTAKAAN

Berisi pembahasan mengenai referensi dan literature mengenai dasar-dasar teori yang digunakan untuk membangun sistem pada penelitian *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering*.

BAB 3 METODOLOGI

Berisi langkah-langkah yang dilakukan dari tahap awal hingga akhir untuk menyelsaikan penelitian ini berupa studi literatur, analisis kebutuhan, perancangan sistem, implementasi metode, pengujian dan penarikan kesimpulan dari penelitian *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering*.

BAB 4 PERANCANGAN

Berisi tahapan dalam perancangan pengujian dan perancangan evaluasi algoritme dari penelitian *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering*.

BAB 5 IMPLEMENTASI

Berisi source code hasil implementasi algoritme untuk penelitian *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering*.

BAB 6 PENGUJIAN DAN ANALISIS

Berisi pembahasan dan analisis hasil pengujian dari penelitian *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering*.

BAB 7 KESIMPULAN DAN SARAN

Berisi penjabaran kesimpulan dan saran yang didapat dari penelitian yang telah dilakukan.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Teori

Kajian teori akan membahas mengenai penelitian-penelitian sebelumnya yang telah dilakukan tentang algoritme *Hierarchical Agglomerative Clustering* dalam menyelesaikan permasalahan *clustering*.

Ms. Shraddha K. Popat melakukan penelitian tentang penerapan algoritme *hierarchical clustering* berdasarkan *cosine similarity* untuk *clustering* dokumen (Popat, Deshmukh, & Metre, 2017). Dalam penelitian tersebut peneliti juga membandingkan metode *hierarchical clustering* dengan metode *clustering* lain yaitu *k-means, centroid similarity*. Nilai *accuracy* pada jumlah klaster 5 untuk metode *HAC* yaitu 0,872, metode *k-means* yaitu 0,708, metode *centroid similarity* yaitu 0,839. Nilai akurasi dari *HAC* selalu tertinggi setiap pertambahan jumlah klaster hingga jumlah klaster 10 dibandingkan dengan metode *K-means* dan *centroid similarity*. Saat jumlah klaster 10 nilai akurasi dari metode *HAC* yaitu 0,77, metode *k-means* yaitu 0,578 dan metode *centroid similarity* yaitu 0,701. Dari penelitian tersebut menghasilkan bahwa metode *hierarchical clustering* memiliki *accuracy* yang lebih tinggi dipandingkan dengan metode lain.

Rendy Handoyo melakukan penelitian tentang perbandingan metode *single linkage* dan *k-means* untuk pengelempokan dokumen. *Single linkage* adalah metode pengelompokan pada *hierarchical clustering* dengan memperhatikan jarak terdekat antar *cluster* (Handoyo, Mangkudjaja, & Nasution, 2014). Dalam penelitian tersebut kualitas *cluster* dihitung menggunakan *Silhouette Coefficient* dan *Purity*. Dari hasil pengujian ketika dokumen berjumlah 100 nilai *silhouette coefficient* untuk metode *single linkage* pada klaster berjumlah 3 adalah 0,04 sedangkan untuk metode *k-means* pada jumlah klaster 3 bernilai 0,01. Nilai *silhouette coefficient* dari metode *single linkage* semakin bertambah ketika jumlah klaster semakin banyak sedangkan untuk metode *k-means* selalu tidak jauh dari nilai 0, Pada saat jumlah klaster 30, nilai *silhouette coefficient* dari *single linkage* adalah 0,24 sedangkan untuk *k-means* bernilai 0. Untuk nilai *purity* dari *single linkage* selalu 1 untuk jumlah klaster mulai dari 3 sampai dengan 30, Sedangkan untuk *k-means* nilai *purity* ketika klaster berjumlah 3 adalah 0,4 dan terus menurun ketika jumlah klaster terus bertambah. Dari penelitian tersebut dihasilkan nilai *Silhouette Coefficient* dari *Single Linkage* selalu lebih unggul dibandingkan dengan *k-means* dan nilai *Purity* dari *Single Linkage* selalu bernilai 1 namun untuk *k-means* tidak pernah bernilai 1. *Single Linkage* selalu menghasilkan dokumen yang sama untuk setiap *cluster*-nya namun untuk *k-means* masih bercampur dengan dokumen lain dalam 1 *cluster*.

Penelitian lain membahas tentang analisis penggunaan jarak untuk *clustering* teks (Sandhya, Lalitha, Govardhan, & Anuradha, 2008). Dalam penelitian tersembut membandingkan *cosine similarity, jaccard coefficient, Euclidean similarity, pearson correlation coefficient*. Perbandingan nilai *entropy* saat menggunakan *vector space* dengan pembobotan *TF-IDF* menghasilkan nilai

terendah untuk *cosine similarity* jika dibandingkan dengan *Euclidean distance.* Pada penelitian ini peneliti menggunakan dataset *Reuters* dan *Classic.* Saat menggunakan dataset *Reuters* nilai *entropy* dari *cosine similarity* dengan menggunakan pembobotan *TF-IDF* adalah 0,36, sedangkan nilai *entropy* dari *Euclidean distance* dengan menggunakan pembobotan *TF-IDF* adalah 0,42. Saat menggunakan dataset *Classic* nilai *entropy* dari *cosine similarity* dengan menggunakan pembobotan *TF-IDF* adalah 0,06 sedangkan nilai *entropy* dari *Euclidean distance* dengan menggunakan pembobotan *TF-IDF* adalah 0,30. Dari penelitian tersebut dapat disimpulkan bahwa *cosine similarity* lebih efektif untuk menyelesaikan permasalahan jarak dalam *vector space* menggunakan *TF-IDF*.

Tabel 2.1 Tabel Kajian Teori

No.	Judul	Penulis	Kajian Teori
1	*Hierarchical Document Clustering based on Cosine Similarity measure*	(Popat, Deshmukh, & Metre, 2017)	Penggunaan *Hierarchical Document Clustering* untuk pengelompokan dokumen dengan *Cosine Similarity* serta perbandingan *Hierarchical Agglomerative Clustering* dengan metode *clustering* lain
2	Perbandingan Metode *Clustering* Menggunakan Metode *Single Linkage* dan *K-Means* Pada Pengelompokan Dokumen	(Handoyo, Mangkudjaja, & Nasution, 2014)	Membandingkan metode *Hierarchical Clustering* menggunakan *Single Linkage* dan metode *K-Means* untuk pengelompokan dokumen, serta pengujian dengan metode *Silhouette Coefficient*
3	*Analysis of Similarity Measures for Text Clustering*	(Sandhya, Lalitha, Govardhan, & Anuradha, 2008)	Analisis perbandingan jarak pada *clustering* dokumen yaitu *cosine similarity, jaccard coefficient, Euclidean similarity* dan *pearson correlation coefficient*

2.2 Dokumen Skripsi

Dokumen skripsi adalah suatu karya ilmiah yang ditulis oleh mahasiswa program S1 yang membahas topik atau bidang tertentu bedasarka hasil kajian pustaka yang ditulis oleh para ahli, hasil penelitian lapangan atau hasil

pengembangan (Huda, 2011). Dokumen skripsi pada akhirnya akan terkumpul menjadi suatu dan dikelola oleh Pergutuan Tinggi melalui fakultas masing-masing. Dokumen skripsi biasanya juga digunakan sebagai referensi oleh mahasiswa lain untuk pengerjaan skripsi maupun penelitian lain. Dokumen skripsi pada studi kasus Fakultas Ilmu Komputer Universitas Brawijaya dikelola oleh 2 tempat yaitu Ruang Baca Fakultas Ilmu Komputer Universitas Brawijaya dan Perpustakaan Pusat Universitas Brawijaya. Pengarsipan dokumen skripsi di Ruang Baca Fakultas Ilmu Komputer Universitas Brawijaya berupa *hardcopy* dikelompokan berdasarkan tahun penerbitan dokumen skripsi. Selain dalam bentuk *hardcopy* juga terdapat *website* yang menampilkan hanya judul-judul skripsi tidak *term*asuk abstrak ataupun isi penuh dari dokumen skripsi tersebut. Dokumen skripsi pada Perpustakaan Pusat Universitas Brawijaya terdapat pengarsipan *hardcopy* dan terdapat *website* untuk menampilkan dokumen-dokumen skripsi seluruh mahasiswa Universitas Brawijaya. Konten yang ditampilkan yaitu judul, abstrak dan beberapa *properties* yang dimiliki oleh dokumen skripsi tersebut seperti Bahasa yang digunakan, program studi yang menerbitkan dan lain-lain.

2.3 *Text Mining*

Text mining merupakan proses pencarian pola pada teks, menganalisis teks untuk diekstrasi informasi yang ada sehingga dapat digunakan untuk tujuan yang bermanfaat (Witten, Frank, & Hall, 2011). Dalam *text mining* terdapat beberapa tahapan untuk mengekstraksi informasi dari kumpulan teks yaitu tokenisasi, *case folding, stemming*, penghilangan *stopword* dan menghitung frekuensi *term*.

2.3.1 Preprocessing

Preprocessing adalah suatu proses yang digunakan untuk mengambil kesimpulan atau mengekstraksi informasi dari suatu sumber data yang tidak terstruktur (Feldman & Sanger, 2007). *Preprocessing* terdapat tahapan-tahapan yaitu tokenisasi, *case folding, stemming,* menghilangkan *stopword* dan menghitung frekuensi dokumen. Tokenisasi adalah proses mengubah kumpulan kata menjadi token-token *term*. Teknik untuk mengubah kalimat menjadi token-token bisa dipisahkan langsung untuk setiap *term* yang dipisahkan oleh karakter *non-alphanumeric*. *Case folding* adalah mengubah kapitalisasi huruf seluruh teks menjadi sama, pada umumnya kapitalisasi huruf diubah menjadi *lower case*. Penyeragaman kapitalisasi huruf ini berfungsi agar tidak terjadi perbedaan *term* yang memiliki kapitalisasi huruf kecil dan besar yang menyebabkan perbedaan perhitungan dalam frekuensi *term. Stemming* adalah proses mengubah seluruh *term* yang ada menjadi kata dasar. Terdapat beberapa algoritme *Stemming* yaitu algoritme Nazief Adriani, Idris, Algoritme Ahmad, Yusoff, dan Sembok, Algoritme Vega, dan Algoritme Arifin dan Setiono. Metode *stemming* yang dilakukan pada penelitian ini yaitu langsung menggunakan *library* dari Sastrawi. *Library* Sastrawi digunakan pada *Python* dengan mengimport *package* Sastrawi. Untuk melakukan *stemming,* setiap *term* yang sudah ditokenisasi dan di-*case folding* dibungkus ke dalam *method stemmer* Sastrawi sehingga langsung menhasilkan *term* yang sudah ter-*stemming.* Proses menghilangkan *stopword* adalah proses menghilangkan kata

atau *term* yang dianggap tidak penting (contoh: dan, kemudian, dengan dan lain-lain). Metode penghilangan *stopword* pada penelitian ini yaitu dengan menggunakan *library* Sastrawi. Setiap *term* dari hasil *Stemming* dibungkus ke dalam *method stopword.remove()* untuk menghapus kata atau *term* yang dianggap tidak penting sesuai dengan daftar kata *stopword* yang dimiliki oleh *library* Sastrawi. Seluruh *term* yang sudah melalui proses tokenisasi, *case folding, stemming* dan penghilangan *stopword* dihitung frekuensi kemunculannya untuk pembobotan kata. *Text mining* dokumen dibutuhkan suatu pembobotan kata sebagai fitur untuk pengolahan kata. Pembobotan yang digunakan adalah pembobotan *tf-id*. Pembobotan *TF-IDF* yaitu memperhatikan *term* yang muncul pada dokumen tertentu.

2.4 Pembobotan *TF-IDF*

Pembobotan *TF-IDF* merupakan pembobotan kata dengan memperhatikan frekuensi kemunculan kata dalam dokumen yang dibentuk ke dalam suatu *vector* yaitu *vector space model* (Saad, Mohamed, & Al-Qutaish, 2012). *Tf* merupakan jumlah kemunculan *term* dalam suatu dokumen. Untuk mendapatkan nilai bobot dari *tf* (Wtf) dapat dihitung dengan menggunakan Persamaan 2.1.

$$Wtf = 1 + \log(tf) \tag{2.1}$$

tf : Frekuensi kemunculan *term* dalam suatu dokumen.

idf adalah *inverse* dari banyaknya dokumen di mana suatu *term* muncul dan *N* adalah jumlah seluruh dokumen. Untuk menghitung nilai *idf* dapat menggunakan Persamaan 2.2.

$$idf = \log(\frac{N}{df}) \tag{2.2}$$

N : Jumlah seluruh dokumen yang digunakan.

df : Jumlah dokumen di mana suatu *term* yang dipilih muncul.

Berikutnya nilai dari bobot *tf* dan *idf* dikalikan untuk menghasilkan bobot *TF-IDF*. Nilai bobot *TF-IDF* dibentuk dalam format matriks baris untuk *term* dan kolom untuk dokumen. Pembobotan *TF-IDF* dapat dihitung menggunakan Persamaan 2.3.

$$TF - IDF = Wtf \times idf \tag{2.3}$$

2.5 *Cosine Similarity*

Cosine Similarity merupakan metode yang digunakan untuk menghitung tingkat kemiripan antara dua data, data pada penilitian disini yaitu dokumen. Perhitungan *cosine similarity* didasarkan pada nilai *vector space model*. Sehingga dalam perhitungan nilai *cosine similarity* yang diperhatikan adalah dua buah dokumen yang dinyatakan dalam dua buah *vector* dengan kata kunci dari suatu dokumen sebagai ukuran (Pradnyana & Sanjaya, 2012).

Nilai *cosine similarity* dapat dihitung menggunakan Persamaan 2.4.

$$CosSim(d_j, q) = \frac{\vec{d_j} \cdot \vec{q}}{\left|\vec{d_j}\right| \cdot \left|\vec{q}\right|} = \frac{\sum_{i=1}^{t}(W_{ij} \cdot W_{iq})}{\sqrt{\sum_{i=1}^{t} {W_{ij}}^2 \cdot \sum_{i=1}^{t} {W_{iq}}^2}} \quad (2.4)$$

$CosSim(d_j, q)$: Hasil nilai *Cosine Similarity* antar 2 dokumen.

d_j : Dokumen yang dipilih.

q : Dokumen pembanding yang dipilih.

$\vec{d_j}$: Panjang vector dokumen yang dipilih.

$\vec{q}$: Panjang vector dokumen pembanding yang dipilih.

W_{ij} : Bobot *TF-IDF* dari dokumen yang dipilih.

W_{iq} : Bobot *TF-IDF* dari dokumen pembanding yang dipilih.

2.5.1 Cosine Distance

Cosine Distance merupakan metode yang digunakan untuk menghitung jarak terdekat antara dua data. Nilai *cosine distance* diperoleh dari 1 dikurangi nilai *cosine similarity* (Sahu & Mohan, 2014). Jangkauan nilai pada *cosine distance* tetap sama dengan *cosine similarity* yaitu antara 1 sampai dengan 0. Nilai dari *cosine distance* dapat dihitung dengan Persamaan 2.5.

$$CosDis(d_j, q) = 1 - CosSim(d_j, q) \quad (2.5)$$

$CosDis(d_j, q)$: Hasil nilai *Cosine Distance* antar 2 dokumen.

$CosSim(d_j, q)$: Nilai *Cosine Similarity* antar 2 dokumen.

2.6 *Clustering*

Clustering adalah metode pengelompokan ketika tidak ada kelas-kelas sebagai target pengelompokan suatu data namun data-data dapat terkelompokan secara natural (Witten, Frank, & Hall, 2011). *Clustering termasuk* dalam kategori *unsupervised* learning yang artinya belajar dari data mentah. *Clustering* mengolah keseluruhan data langsung yang mana data tersebut tidak memiliki label kelas. Berbeda dengan klasifikasi di mana metode belajar dari data latih yang memiliki label kelas kemudian terdapat data uji yang diolah dan dikelompokan sesuai label kelas pada data latih. Tujuan dari *clustering* dokumen yaitu apabila membutuhkan pencarian terkait informasi dokumen menghasilkan informasi yang sesuai dengan kebutuhan sehingga pengguna juga lebih cepat dalam menemukan informasi.

Pada dasarnya *clustering* merupakan suatu analisis untuk kumpulan data untuk mengetahui beberapa data masuk dalam kelompok tertentu yang memiliki tingkat kemiripan tertentu. Tingkat kemiripan suatu dokumen dapat dihitung menggunakan beberapa metode diantaranya *cosine similarity, Euclidean distance* dan lain-lain. *Cosine similarity* digunakan untuk mengukur tingkat kemiripan dokumen, semakin tinggi nilai *cosine similarity* berarti semakin tinggi tingkat

kemiripan dokumen begitu juga sebaliknya. *Euclidean distance* merupakan metode untuk mendapatkan jarak dari suatu dokumen, semakin tinggi nilai *Euclidean distance* berarti tingkat kemiripan suatu dokumen semakin jauh. Dalam penelitian ini digunakan metode *cosine distance* yaitu nilai jarak yang didapat dari nilai 1 dikurangi nilai *cosine similarity*.

Algoritme *clustering* terbagi menjadi 2 metode yaitu *flat clustering* dan *hierarchical clustering*. *Flat clustering* adalah *clustering* suatu data dengan terlebih dahulu ditentukan jumlah *cluster* dan *centroid*-nya. Sehingga seluruh data diolah untuk dikelompokan ke dalam *cluster* yang sudah ditentukan dengan menghitung kemiripan dengan *centroid* yang telah ditentukan. *Hierarchical clustering* merupakan algoritme *clustering* yang tidak memerlukan penentuan jumlah *cluster* dan *centroid*. Dalam *hierarchical clustering* terdapat 2 pendekatan yaitu *agglomerative* dan *divisive*. *Agglomerative* merupakan pendekatan *clustering* dimulai dari data berupa dokumen yang bersifat individu atau *cluster* yang kecil kemudian bertahap *cluster* demi *cluster* digabungkan sehingga menjadi 1 *cluster* besar. *Divisive* merupakan kebalikan dari *agglomerative* yaitu 1 *cluster* besar dari dokumen kemudian dipisah secara bertahap menjadi beberapa *cluster* kecil. Pada penelitian ini digunakan metode *hierarchical agglomerative clustering*.

2.7 *Hierarchical Agglomerative Clustering (HAC)*

Hierarchical Agglomerative Clustering adalah salah satu teknik yang digunakan untuk *clustering* dokumen. Dalam *Hierarchical Agglomerative Clustering* dokumen dikelompokan secara berulang berdasarkan kemiripan yang terdekat antar data yang didasarkan dari pembobotan kata (Jaiswal & Janwe, 2011). Metode ini mengelompokkan suatu data dimulai dari data per-individu yang kemudian dikelompokan hingga membentuk grup-grup dan proses pengelompokan secara berulang untuk menggabungkan grup-grup yang memiliki kemiripan hingga seluruh data terkelompokkan, sehingga proses ini biasa disebut dengan pendekatan *bottom-up*. Dalam prosesnya *Hierarchical Agglomerative Clustering* tidak memerlukan inisialisasi jumlah *cluster* dan inisialisasi *centroid* seperti halnya pada *flat clustering*. Dalam mengelompokkan suatu dokumen dibutuhkan suatu acuan. Acuan yang digunakan pada penelitian ini untuk mengelompokkan dokumen yaitu dengan jarak terdekat antar dokumen. Jarak terdekat antar dokumen dapat dihitung menggunakan *cosine distance*. Setelah nilai *distance* tiap dokumen diperoleh berikutnya terdapat metode untuk pengambilan nilai yang digunakan untuk mengelompokkan dokumen. Terdapat beberapa metode pengambilan nilai untuk pengelompokan dokumen yaitu diantaranya adalah *single linkage, complete linkage, average linkage*. Dalam penelitian ini metode pengelompokan dokumen berdasarkan 3 nilai jarak sebagai parameter yaitu menggunakan *Single linkage, Complete Linkage* dan *Average Linkage*. *Single linkage* merupakan metode pengelompokan dokumen berdasarkan nilai *cosine distance* atau jarak terdekat dari 2 data dari 2 *cluster* berbeda. *Complete linkage* merupakan metode pengelompokan dokumen dengan *update* nilai *cosine distance* atau jarak dari 2 dokumen dengan melihat jarak terjauh dari 2 data dari 2 *cluster* berbeda. *Average linkage* merupakan metode pengelompokan dokumen

dengan *update* nilai *cosine distance* atau jarak dari 2 dokumen dengan melihat jarak rata-rata dari 2 data dari 2 *cluster* berbeda. Dokumen-dokumen yang telah terkelompokan ke dalam *cluster-cluster* tertentu berikutnya *cluster-cluster* akan dikelompokan kembali hingga membentuk 1 *cluster* besar. Proses pengelompokan dari individu dokumen menjadi *cluster-cluster* kecil hingga 1 *cluster* besar pada dasarnya membentuk suatu pohon. Dalam *Hierarchical Agglomerative Clustering* setiap tahapan pengelompokkan akan disimpan untuk mengetahui *cluster* yang terbentuk atau dapat menggunakan *dendrogram* untuk memperjelas proses pembentukkan *cluster* yang berbentuk menyerupai pohon.

2.7.1 Proses *Hierarchical Agglomerative Clustering*

1. Menghitung frekuensi kemunculan suatu *term* dalam dokumen *(tf)*.
2. Menghitung bobot frekuensi kemunculan *term* pada suatu dokumen *(Wtf).* Bobot frekuensi *term* dapat dihitung dengan Persamaan 2.1.
3. Menghitung jumlah dokumen yang mana suatu *term* yang dipilih muncul pada suatu dokumen *(df).*
4. Menghitung nilai *invers* dari jumlah dokumen yang mana suatu *term* yang dipilih muncul pada suatu dokumen *(idf).* Nilai *invers* dokumen dapat dihitung dengan Persamaan 2.2.
5. Menghitung bobot *TF-IDF* dengan mengalikan nilai *Wtf* dengan *idf.*
6. Melakukan normalisasi terhadap bobot *TF-IDF.* Normalisasi bobot *TF-IDF* dapat dilakukan dengan Persamaan 2.4.
7. Menghitung nilai jarak antar dokumen dibentuk dalam suatu table *Cosine Distance*. Nilai *Cosine Similarity* dapat dihitung dengan Persamaan 2.5.
8. 2 dokumen/*cluster* dengan nilai *Cosine Distance* terendah akan digabung menjadi 1 kelompok. Penggabungan 2 dokumen menggunakan teknik *Single Linkage/Complete Linkage/Average Linkage*.
9. Menghitung kembali nilai *Cosine Distance* dengan melibatkan kelompok dokumen baru. Nilai *cosine distance* dari kelompok dokumen baru merupakan nilai *cosine distance* terkecil diantara 2 dokumen yang dikelompokan.
10. Mengulangi kembali langkah ke 7.

2.8 Pelabelan *Cluster*

Pelabelan *cluster* adalah proses memberikan identitas berupa nama pada suatu *cluster* agar *cluster* tersebut dapat dikenali. Nama yang digunakan sebagai label dari suatu *cluster* merupakan kata yang mewakili isi dari *cluster.* Proses pelabelan suatu *cluster* menggunakan metode pembobotan *TF-IDF.* Detail perhitungan dari pembobotan *TF-IDF* dapat dilihat pada Sub-Bab 2.4. Tahapan yang dilakukan adalah yang pertama yaitu menghitung bobot *TF-IDF term* pada setiap *cluster* kemudian melakukan pengurutan berdasarkan bobot tertinggi dari

term pada setiap *cluster* (Nunes, et al., 2014). Hasil dari pengurutan *term* diambil beberapa *term* dengan bobot tertinggi sebagai label dari suatu *cluster.*

2.9 Evaluasi

Evaluasi berupa pengujian digunakan untuk mengetahui seberapa baik klaster yang dihasilkan dari metode *Hierarchical Agglomerative Clustering* dan evaluasi untuk ketepatan label *cluster* yang diperloleh. Metode pengujian metode *Hierarchical Agglomerative Clustering* yang digunakan adalah *Silhouette Coefficient* dan untuk evaluasi ketepatan label *cluster* menggunakan *Precision.*

2.9.1 Silhouette Coefficient (SC)

Silhouette Coefficient (SC) merupakan metode evaluasi yang digunakan untuk melihat kualitas dan kekuatan *cluster* dan kualitas suatu objek dalam suatu *cluster* (Handoyo, Mangkudjaja, & Nasution, 2014). Untuk menghitung *silhouette coefficient* terdapaat beberapa tahapan yaitu:

1. Menghitung rata-rata jarak setiap dokumen (disimbolkan dalam *i*) dengan setiap dokumen lain dalam *cluster* yang sama. Simpan nilai rata-rata pada suatu variable misalkan a_i.
2. Menghitung rata-rata nilai jarak setiap dokumen suatu *cluster* (disimbolkan dalam *i*) dengan setiap dokumen pada *cluster* lain dan diambil nilai terkecil. Simpan nilai tersebut pada suatu variable misalkan b_i.
3. Menghitung nilai *silhouette coefficient* dengan Persamaan 2.6.

$$SC = \frac{b_i - a_i}{\max(a_i, b_i)} \tag{2.6}$$

Nilai *silhouette coefficient* bervariasi antara -1 hingga 1. *Cluster* dikatakan baik apabila nilai *Silhouette Coefficient* bernilai positif. Apabila nilai *Silhouette Coefficient* = 1 artinya dokumen pada *cluster i* berada pada *cluster* yang tepat. Apabila *Silhouette Coefficient* = 0 artinya dokumen pada *cluster i* masih bisa dianggap masuk pada *cluster* lain sehingga dokumen *i* posisinya tidak jelas. Jika *Silhouette Coefficient* = -1 artinya *cluster* mengalami *overlapping* dan dokumen *i* lebih cocok untuk masuk ke *cluster* lain. Nilai *silhouette coefficient* untuk *singleton* atau *cluster* yang didalamnya hanya terdapat 1 data atau dokumen adalah 0 (Rousseeuw, 1987).

2.9.2 Precision

Precision merupakan tingkat ketepatan antara informasi yang diminta oleh *user* dengan hasil jawaban yang diberikan oleh sistem (Feldman & Sanger, 2007). Untuk menghitung *precision*dapat menggunakan Persamaan 2.7.

$$Precision = \frac{A}{B} \tag{2.7}$$

A: Jumlah label yang diperoleh dan relevan terhadap *ground truth*.

B: Jumlah seluruh label yang diperoleh.

BAB 3 METODOLOGI PENELITIAN

Bab ini menjelaskan mengenai alur metodologi penelitian. Metodologi penelitian digunakan sebagai dasar pedoman dalam pelaksanaan penelitian.

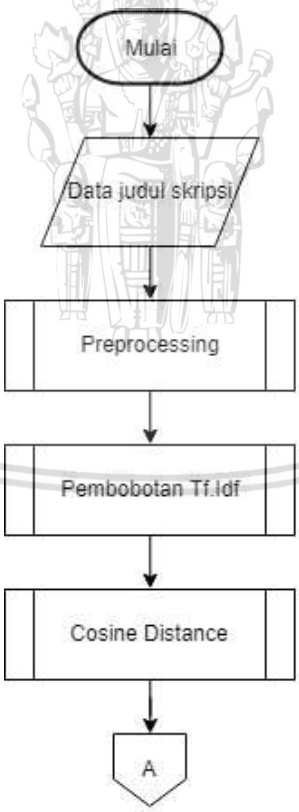
3.1 Tipe Penelitian

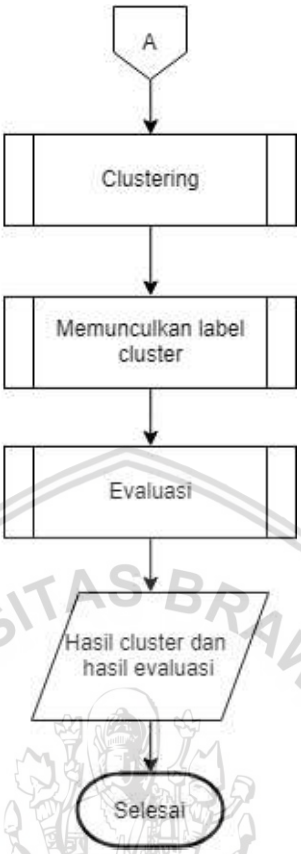
Tipe penelitian yang digunakan pada penelitian adalah tipe penelitian non-implementatif analitik. Kegiatan penelitian yang dilakukan adalah untuk menganalisis derajat hubungan antar elemen dalam objek yang diteliti. Dalam penelitian ini yaitu menganalisis *cluster* yang terbentuk dari pengelompokkan dokumen skripsi.

3.2 Strategi dan Rancangan Penelitian

Strategi dan rancangan penelitian yang dilakukan akan dijelaskan dalam Sub-Bab berikut.

3.2.1 Metode Secara Umum





Gambar 3.1 Diagram alir metode *Hierarchical Agglomerative Clustering*

Dalam penelitian ini akan metode *Hierarchical Agglomerative Clustering* untuk *clustering* dokumen skripsi berdasarkan data judul skripsi. Data masukkan adalah data dokumen skripsi bagian judul. Langkah pertama untuk pengolahan data yaitu tahap *preprocessing.* Langkah detail dari *preprocessing* dapat dilihat pada Sub-Bab 2.3.1. Data hasil *preprocessing* akan masuk ke tahap pengolahan data menggunakan metode *Hierarchical Agglomerative Clustering* yang langkah detailnya dapat dilihat pada Sub-Bab 2.7.1. Jarak atau *linkage* sebagai parameter pengelompokkan yang digunakan adalah *Single Linkage, Complete Linkage* dan *Average Linkage*. Hasil dari pengolahan dokumen ini adalah berupa *cluster-cluster* dari dokumen skripsi dan juga label yang dihasilkan dari *cluster* yang ada. Hasil *clustering* berikutnya akan dievaluasi dengan menggunakan *Silhouette Coefficient.* Tahapan *Silhouette Coefficient* dapat dilihat pada Sub-Bab 2.8.1.

3.2.2 Objek Penelitian

Objek yang disertakan dalam penelitian ini adalah dokumen skripsi Teknik Informatika Universitas Brawijaya. Bagian dokumen skripsi yang digunakan sebagai data adalah judul skripsi. Jumlah dokumen yang digunakan sebanyak 100 dokumen.

3.2.3 Metode Pengumpulan Data

Dokumen skripsi Teknik Informatika Universitas Brawijaya diambil dari *website* Perpustakaan Pusat Universitas Brawijaya http://digilib.ub.ac.id:81/bkg/. Setiap judul yang diambil dipindahkan ke file *CSV.* Jumlah data yang diambil sebanyak 100 dokumen secara acak dengan kata kunci pencarian "Teknik Informatika". Data diambil dari bulan Februari 2018 sampai dengan bulan Juni 2018.

3.2.4 Peralatan Pendukung

Penelitian ini dilakukan pada computer dengan spesifikasi sebagai berikut:

- Processor : Intel® Core™ i5-7200U CPU @ 2.50GHz
- RAM : 8.00 GB
- OS : Windows 10 Pro 64-bit

Bahasa pemrograman yang digunakan untuk mengimplementasikan metode dalam penelitian ini adalah Python 2.7.

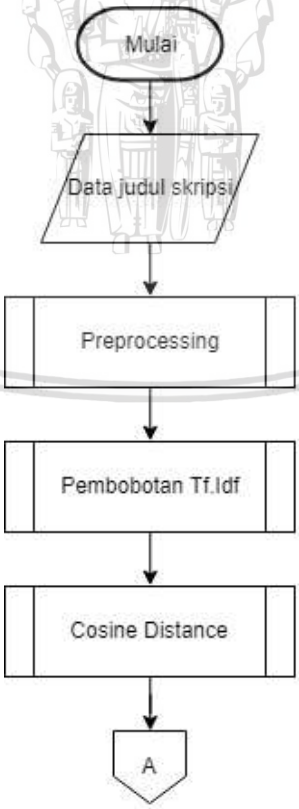
BAB 4 PERANCANGAN

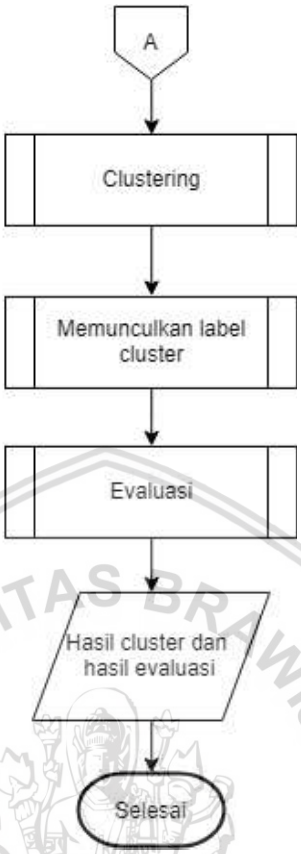
4.1 Formulasi Permasalahan

Permasalahan yang akan diselesaikan pada penilitian ini adalah *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering.* Data yang digunakan adalah data yang berasal dari *website* Perpustakaan Pusat Universitas Brawijaya yaitu http://digilib.ub.ac.id:81/bkg/. Jumlah data yang diambil sebanyak 100 dokumen secara acak dengan kata kunci pencarian "Teknik informatika" untuk mendapatkan dokumen skripsi program studi Teknik informatika. Bagian dokumen yang diambil untuk diolah adalah bagian judul skripsi. Hasil *clustering* akan dievaluasi menggunakan metode *Silhouette Coefficient* untuk melihat ketepatan suatu dokumen pada suatu *cluster.*

4.2 Penyelesaian Permasalahan

Metode *Hierarchical Agglomerative Clustering (HAC)* digunakan untuk menyelesaikan permasalahan diatas dengan tujuan dapat mengelompokkan dokumen pada *cluster* yang tepat. Proses penyelesaian permasalahan menggunakan metode *HAC* secara garis besar dapat dilihat pada Gambar 4.1.





Gambar 4.1 Diagram alir metode *Hierarchical Agglomerative Clustering*

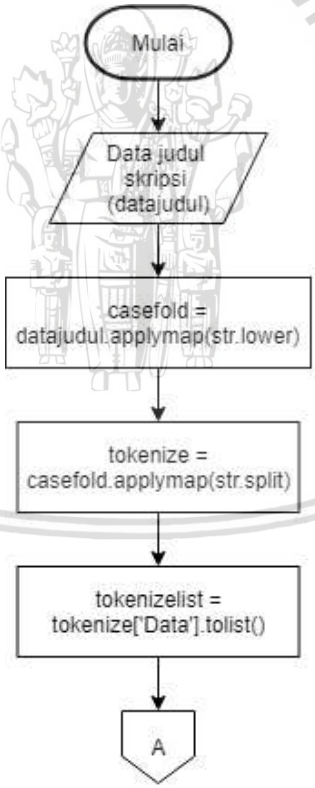
Berdasarkan Gambar 4.1, langkah-langkah metode *HAC* dalam mengelompokkan dokumen skripsi adalah sebagai berikut:

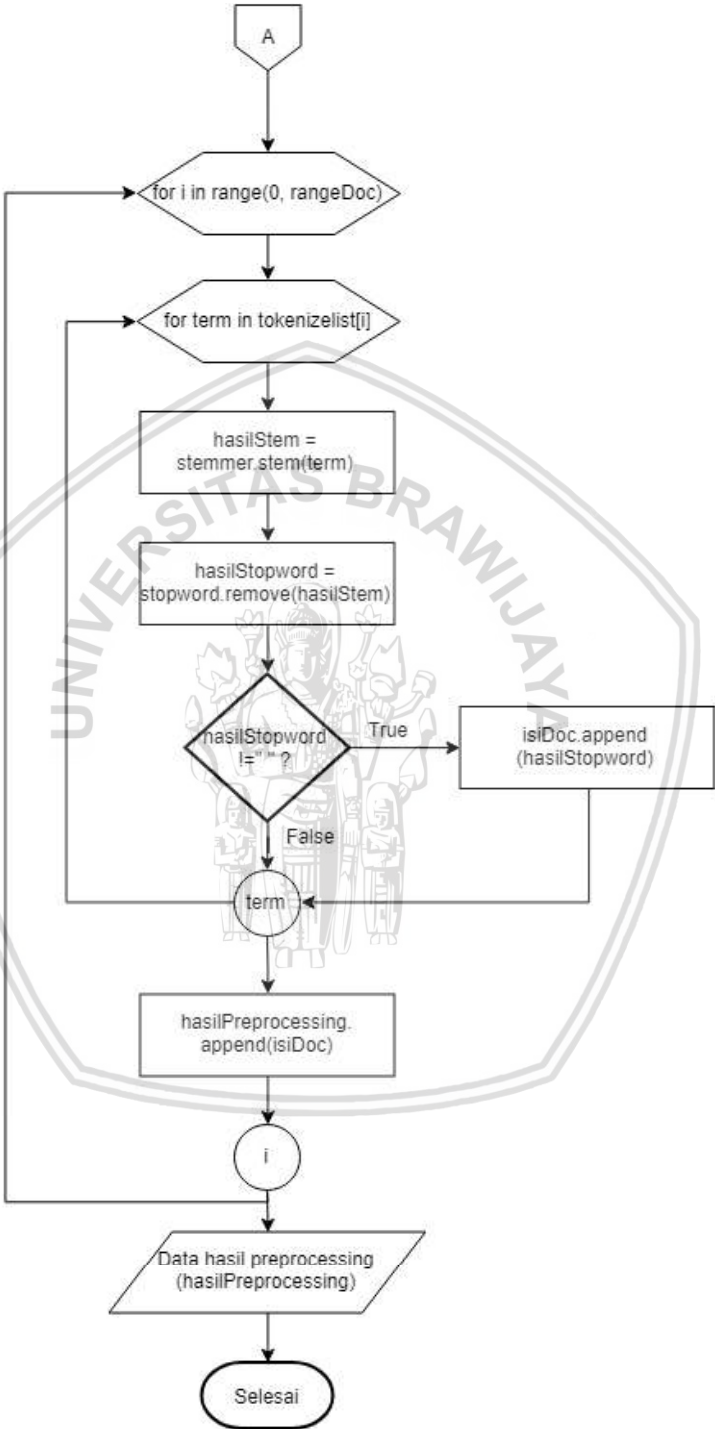
1. Memasukkan data dokumen skripsi berupa judul dan abstrak skripsi.
2. Melakukan tahap *preprocessing* terhadap judul dan abstrak skripsi.
3. Melakukan pembobotan *TF-IDF* terhadap *term* yang dihasilkan dari tahap *preprocessing.*
4. Melakukan normalisasi terhadap nilai bobot *TF-IDF* yang dihasilkan.
5. Melakukan perhitungan *Cosine Distance* untuk mengetahui nilai jarak tiap dokumen.
6. Melakukan penggabungan tiap dokumen atau *cluster* dengan metode *single linkage* yaitu tiap dokumen atau *cluster* yang memiliki jarak terdekat.
7. Jika seluruh dokumen atau *cluster* belum terkelompokkan maka kembali ke proses penggabungan dokumen atau *cluster* menggunakan *single linkage,* jika seluruh dokumen atau *cluster* sudah terkelompokkan maka menuju proses selanjutnya.

8. Mengambil jumlah *cluster* yang dihasilkan dengan cara memotong sumbu Y dari visualisasi dendrogram yang dihasilkan.
9. Memunculkan label dari tiap *cluster* yang dihasilkan.
10. Melakukan tahap evaluasi dari *cluster* yang dihasilkan menggunakan *Silhouette Coefficient.*
11. Hasil akhir yang didapat adalah berupa *cluster* dari kumpulan dokumen skripsi dan nilai tingkat ketepatan dari *cluster* dalam mengelompokkan dokumen.

4.2.1 *Preprocessing* Data

Preprocessing data merupakan proses yang digunakan untuk menyeleksi teks pada dokumen dengan tujuan mengambil kesimpulan atau mengekstraksi informasi dari dokumen yang diproses. Gambar 4.2 menunjukkan langkah-langkah *preprocessing* data yang dilakukan.





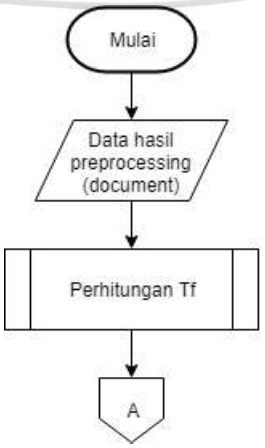
Gambar 4.2 Diagram alir *Preprocessing*

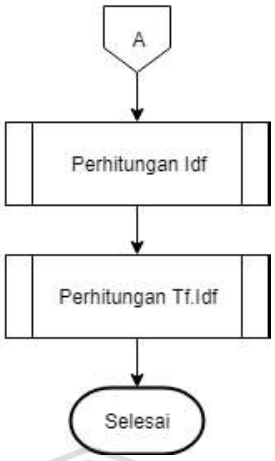
Penjelasan tahapan-tahapan pada diagram alir *preprocessing:*

1. Data judul dan abstrak skripsi yang berformat *csv* dimasukkan ke proses *preprocessing.*
2. Mengubah kapitalisasi huruf pada data judul dan abstrak skripsi menjadi *lower case* menggunakan fungsi *lower.*
3. Proses tokenisasi yaitu memecah suatu teks menjadi token-token *term* menggunakan fungsi *split.*
4. Menyimpan hasil tokenisasi dari judul skripsi ke dalam *list*.
5. Menyimpan hasil tokenisasi dari abstrak skripsi ke dalam *list*.
6. Menggabungkan *list* hasil tokenisasi judul dan *list* hasil tokenisasi abstrak.
7. Untuk setiap *term* pada judul dan abstrak tiap dokumen skripsi dilakukan proses *stemming. Stemming* yang dilakukan menggunakan *library* Sastrawi sehingga langsung dapat dipanggil dengan fungsi *Stemmer.stem(term).*
8. Menghilangkat *term* yang dianggap tidak penting atau disebut *stopword* menggunakan *library* Sastrawi. Penghilangan *stopword* dilakukan dengan langsung menggunakan fungsi *stopword.remove(hasilStem), hasilStem* adalah variable yang berisi *term* hasil proses *stemming.*
9. Memasukkan hasil penghilangan *stopword* ke suatu *list.*
10. Hasil keluaran dari proses ini yaitu berupa *list* yang berisi *term* hasil *preprocessing.*

4.2.2 Pembobotan *TF-IDF*

Pembobotan *TF-IDF* merupakan pembobotan kata dengan memperhatikan frekuensi kemunculan kata dalam dokumen yang dibentuk ke dalam suatu *vector* yaitu *vector space model* dan menghitung *invers document frequency.* Detail perhitungan pombobotan *TF-IDF* dapat dilihat pada Sub-Bab 2.4. Gambar 4.3 menunjukkan langkah-langkah pembobotan *TF-IDF* yang dilakukan.





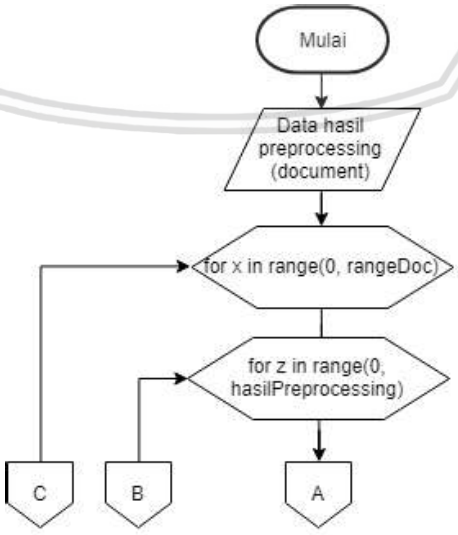
Gambar 4.3 Diagram alir pembobotan *TF-IDF*

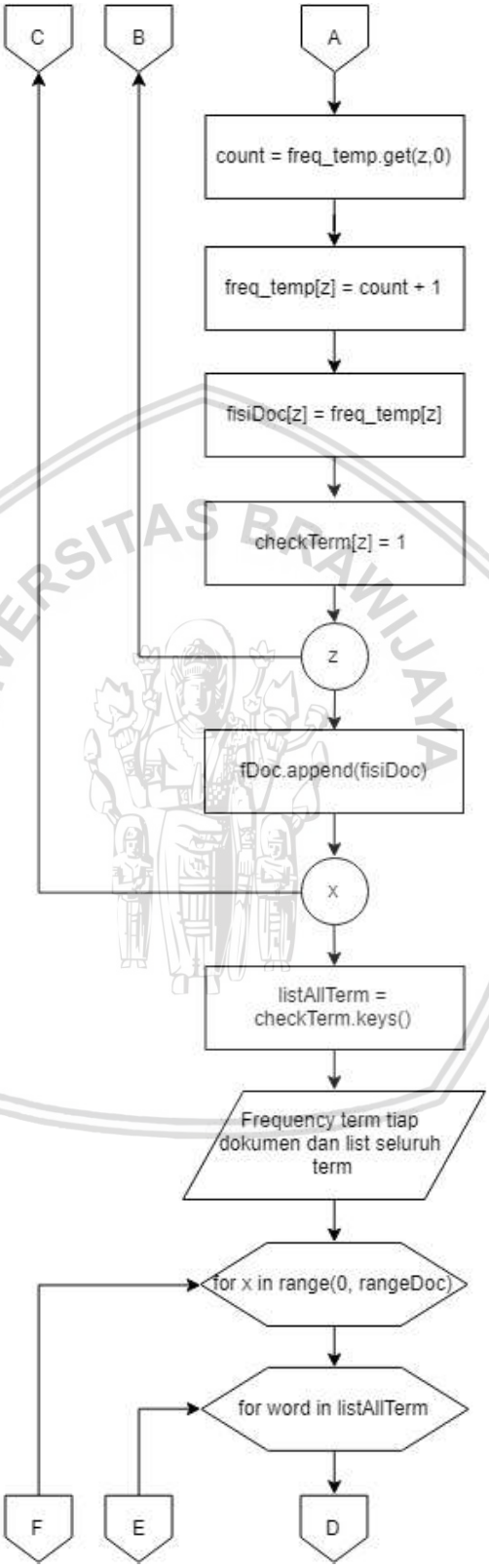
Tahapan-tahapan yang dilalui pada proses pembobotan *TF-IDF* adalah:

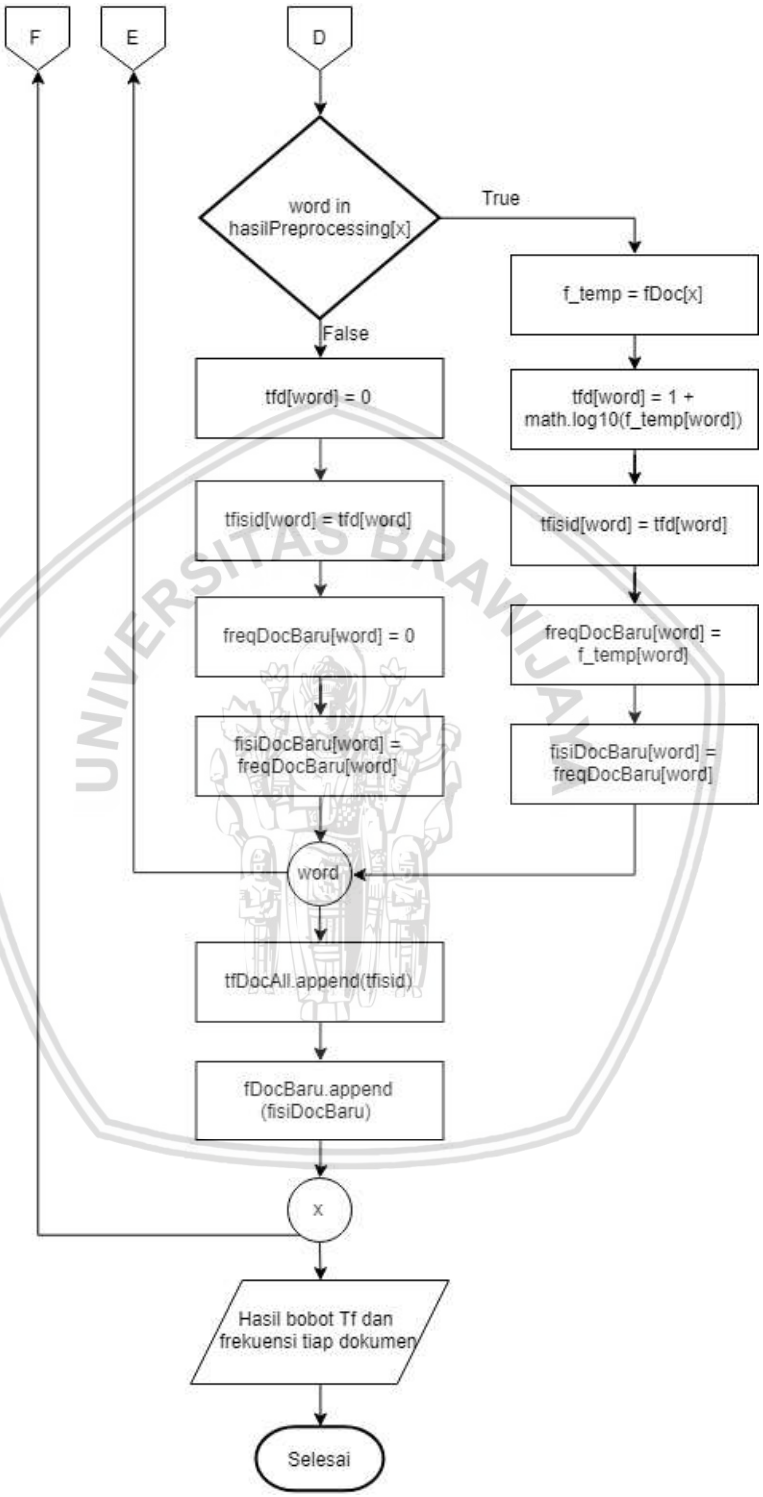
1. Memasukkan data hasil dari proses *preprocessing* berupa *list.*
2. Menghitung nilai bobot kemunculan suatu *term* pada suatu dokumen.
3. Menghitung jumlah dokumen di mana suatu *term* muncul *(idf).*
4. Menghitung bobot *TF-IDF* dengan mengalikan nilai *idf* dengan tiap-tiap bobot *tf.*
5. Keluaran yang dihasilkan adalah nilai *TF-IDF* dari seluruh *term* yang muncul pada tiap-tiap dokumen.

4.2.2.2 Perhitungan Tf

Tf merupakan jumlah kemunculan *term* dalam suatu dokumen. Perhitungan *tf* dilakukan untuk menghasilkan bobot *tf* untuk tiap-tiap *term* dalam dokumen. Langkah-langkah untuk perhitungan *tf* dapat dilihat pada Gambar 4.4.




C
B
A
count = freq_temp.get(z,0)
freq_temp[z] = count + 1
fisiDoc[z] = freq_temp[z]
checkTerm[z] = 1
z
fDoc.append(fisiDoc)
x
listAllTerm = checkTerm.keys()
Frequency term tiap dokumen dan list seluruh term
for x in range(0, rangeDoc)
for word in listAllTerm
F
E
D

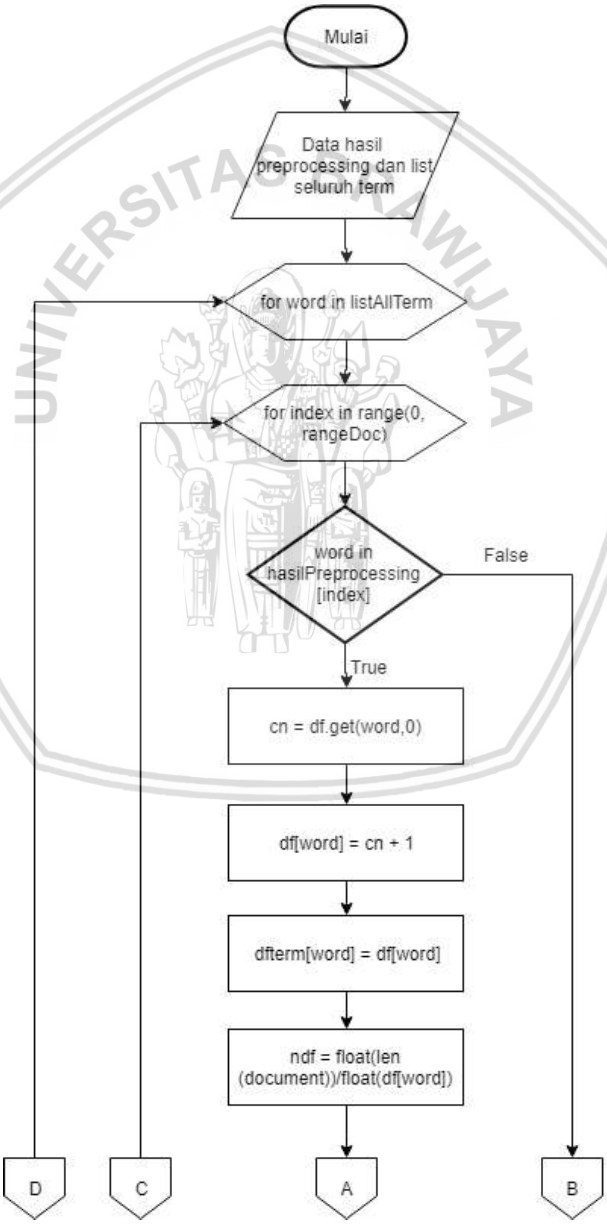


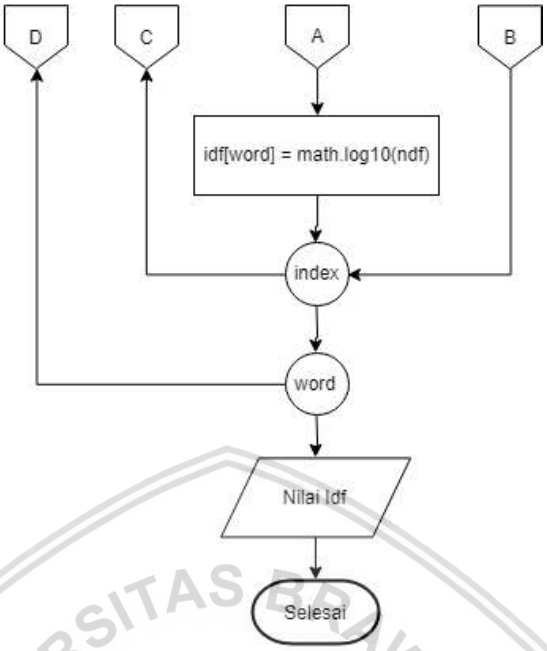
Gambar 4.4 Diagram alir perhitungan *Tf*

Data hasil *preprocessing* dalam bentuk *list* diolah untuk dihitung frekuensi kemunculan suatu *term* dalam suatu dokumen. Nilai frekuensi kemunculan suatu *term* dalam dokumen selanjutnya dihitung untuk menghasilkan bobot *Tf.* Detail rumus perhitungan *Tf* dapat dilihat pada Sub-Bab 2.4 Persamaan 2.1. Keluaran yang dihasilkan yaitu nilai bobot *Tf* seluruh *term* yang muncul pada tiap-tiap dokumen. Pada proses perhitungan *Tf* ini juga menghasilkan *list* frekuensi seluruh *term* yang muncul dari gabungan seluruh data judul dan abstrak skripsi.

4.2.2.3 Perhitungan *idf*

idf adalah *inverse* dari banyaknya dokumen di mana suatu *term* muncul. Langkah-langkah untuk perhitungan *idf* dapat dilihat pada Gambar 4.5.



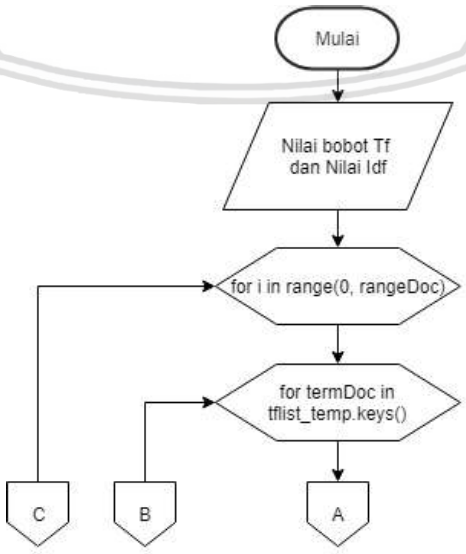


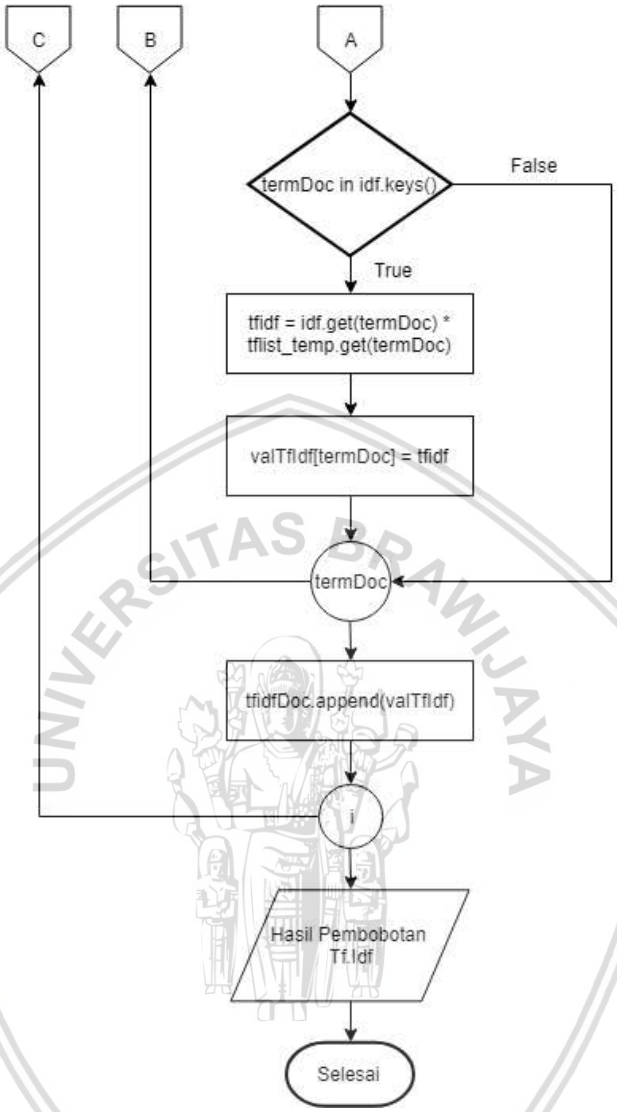
Gambar 4.5 Diagram alir perhitungan *idf*

Masukan dari proses ini adalah data dokumen hasil *preprocessing* dan *list* frekuensi seluruh *term* yang muncul. Proses ini akan menghitung jumlah dokumen di mana suatu *term* muncul *(df).* Dari nilai *df* yang dihasilkan selanjutnya dihitung *inverse document frequency*-nya. Detail rumus perhitungan dari *idf* dapat dilihat pada Sub-Bab 2.4 Persamaan 2.2. Keluaran yang dihasilkan dari proses ini yaitu nilai *idf* dari semua *term* yang muncul dari gabungan seluruh dokumen yang diproses.

4.2.2.4 Perhitungan *TF-IDF*

Langkah-langkah untuk perhitungan *TF-IDF* dapat dilihat pada Gambar 4.6.



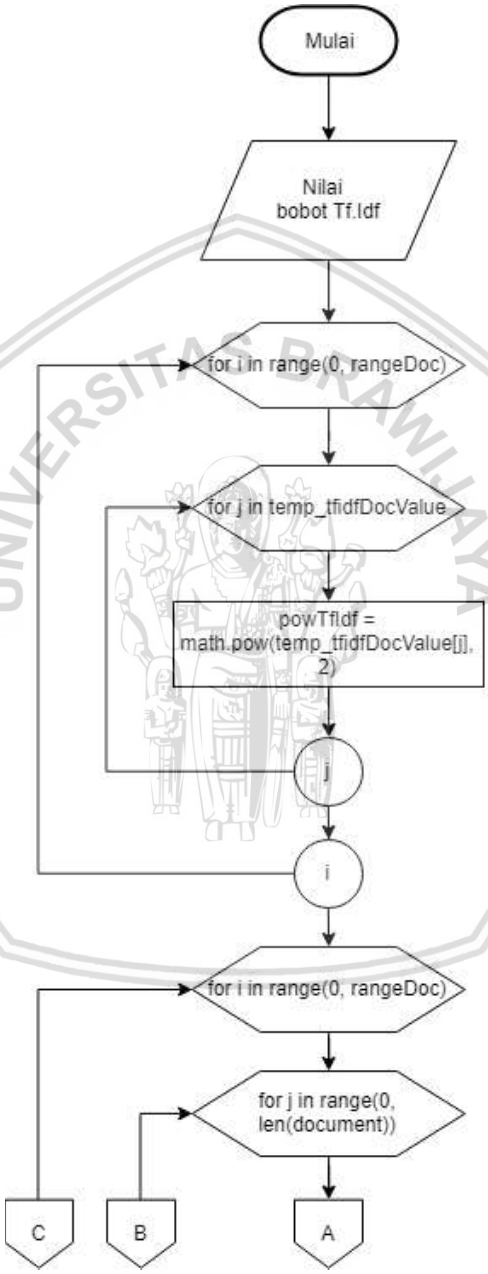


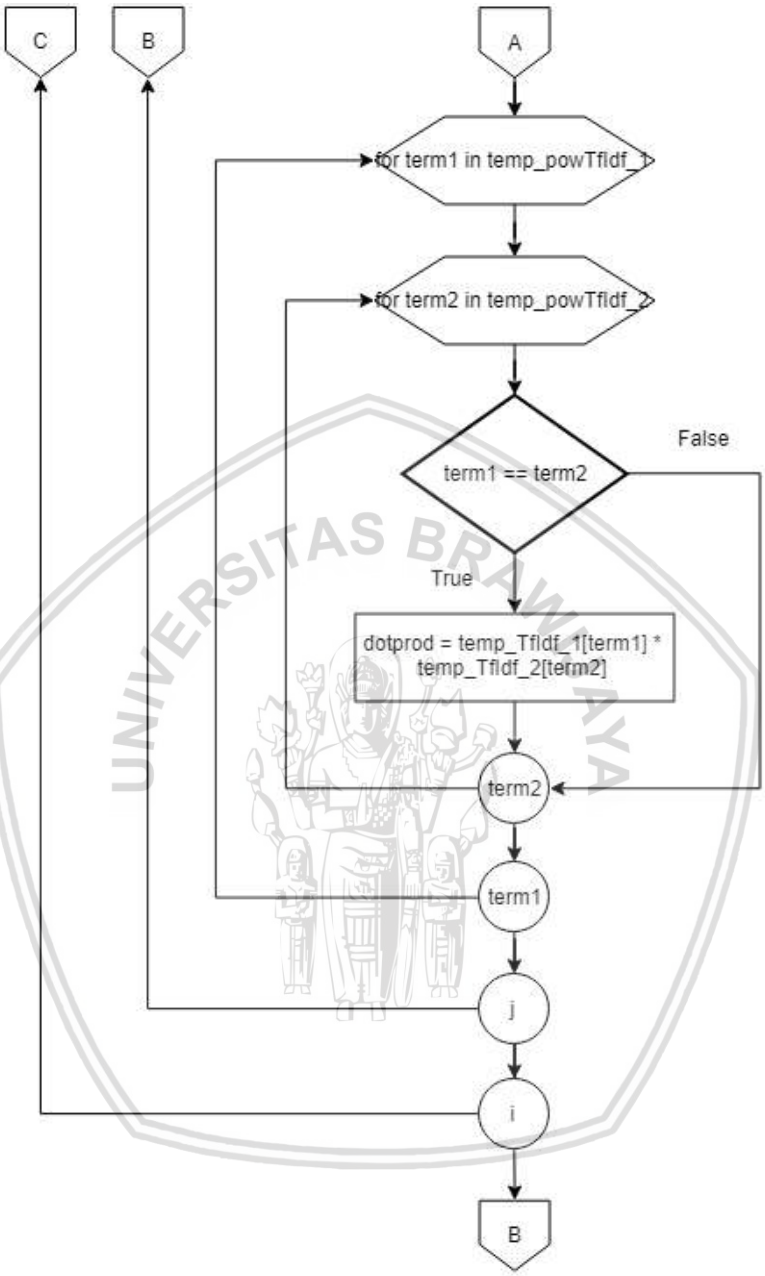
Gambar 4.6 Diagram alir perhitungan *TF-IDF*

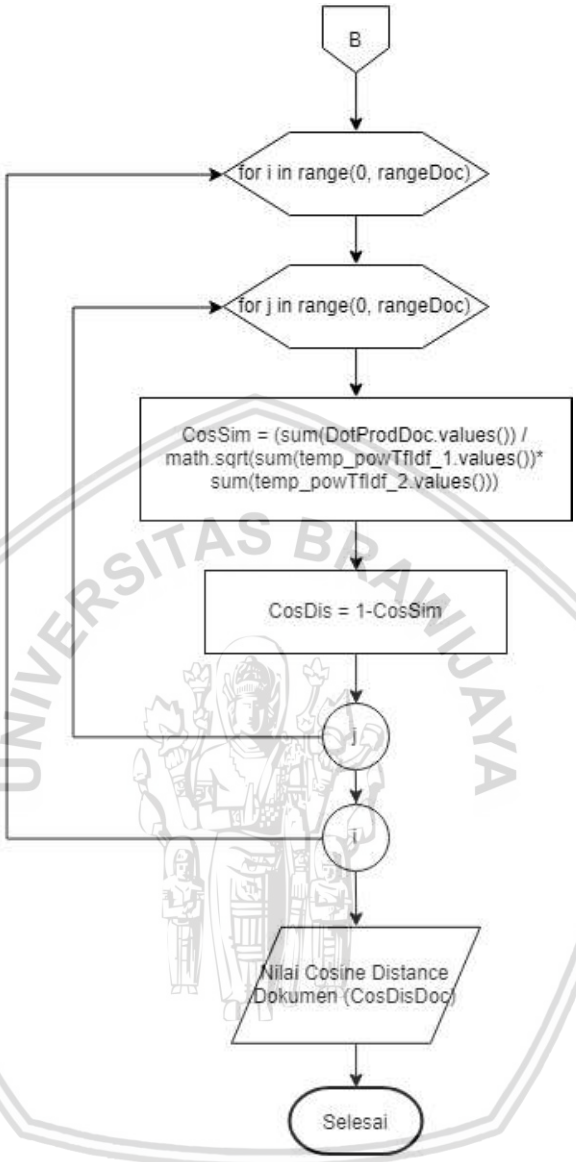
Masukan dari proses perhitungan *TF-IDF* adalah nilai *tf* dari seluruh *term* dari tiap-tiap dokumen dan nilai *idf* dari semua *term* yang muncul dari gabungan seluruh dokumen. Proses perhitungan *TF-IDF* adalah mengalikan nilai *tf* dan *idf.* Detail rumus perhitungan dari *TF-IDF* dapat dilihat pada Sub-Bab 2.4 Persamaan 2.3. Keluaran yang dihasilkan pada proses ini yaitu nilai bobot *TF-IDF* seluruh *term* yang muncul pada tiap-tiap dokumen.

4.2.3 *Cosine Distance*

Cosine Distance merupakan metode yang digunakan untuk menghitung nilai jarak antara dua data, data pada penilitian disini yaitu dokumen. Langkah-langkah untuk perhitungan *cosine distance* dapat dilihat pada Gambar 4.7.




C
B
A
for term1 in temp_powTfIdf_1
for term2 in temp_powTfIdf_2
term1 == term2
False
True
dotprod = temp_TfIdf_1[term1] *
temp_TfIdf_2[term2]
term2
term1
j
i
B

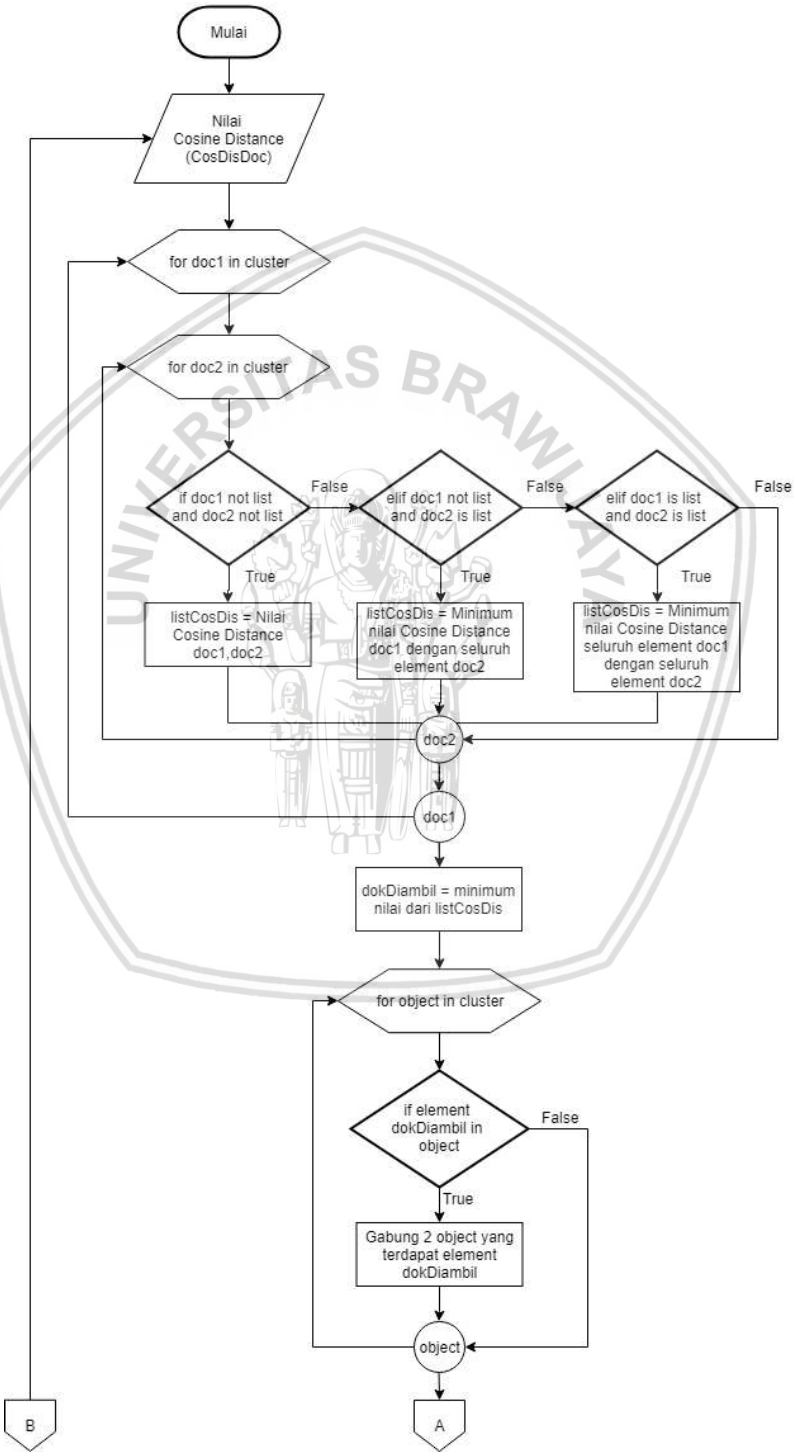


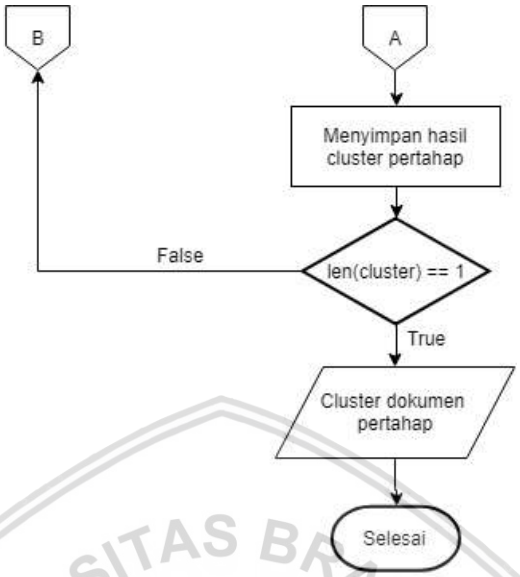
Gambar 4.7 Diagram alir perhitungan *Cosine Distance*

Masukan dari proses perhitungan *cosine distance* adalah nilai bobot *TF-IDF* seluruh *term* dari tiap-tiap dokumen. *Cosine distance* menghitung *dot product* dari 2 dokumen. Masing-masing nilai normalisasi bobot *TF-IDF* dari *term* yang sama dari 2 dokumen dikalikan kemudian nilai *cosine distance* didapat dari 1 dikurangi jumlah seluruh hasil kali tiap *term* antar 2 dokumen. Detail rumus perhitungan dari *cosine similarity* dapat dilihat pada Sub-Bab 2.5.1 Persamaan 2.6. Hasil keluaran dari proses ini adalah nilai *cosine distance* dari kombinasi 2 dokumen dari seluruh dokumen yang ada

4.2.4 Penggabungan dengan *Single Linkage*

Penggabungan 2 dokumen dengan *Single Linkage* merupakan tahap *clustering* dengan memperhatikan jarak (nilai *Cosine Distance)* terdekat atau terkecil. Langkah-langkah untuk penggabungan dokumen atau *cluster* dengan *single linkage* dapat dilihat pada Gambar 48.



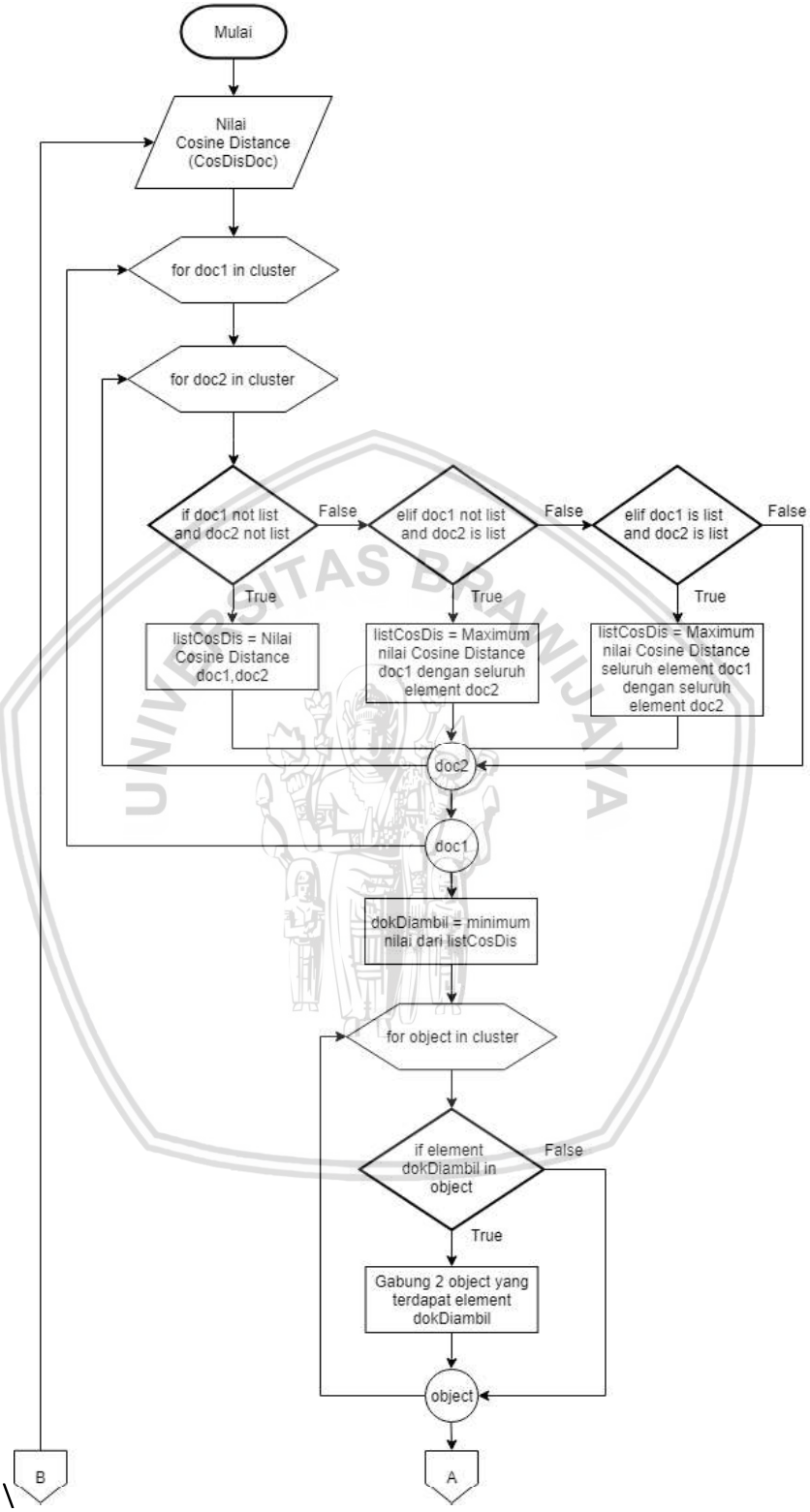


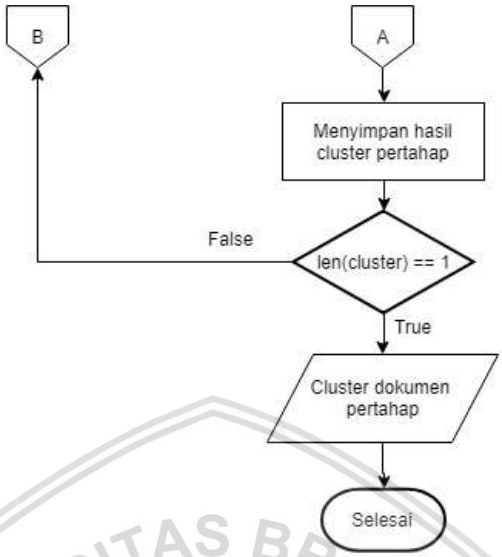
Gambar 4.8 Diagram alir penggabungan data dengan *Single Linkage*

Masukan pada proses penggabungan data ini yaitu nilai *cosine distance* dari tiap-tiap dokumen dengan dokumen lain. Metode *single linkage* akan menggabungkan 2 dokumen atau 2 kelompok data dengan nilai *cosine distance* tertendah. Setelah 2 dokumen atau 2 kelompok data digabungkan selanjutnya akan meng-*update* kembali nilai *cosine distance* dengan melibatkan 2 dokumen atau 2 kelompok data. Jika menggunakan *single linkage* maka nilai *cosine distance* dari 2 dokumen atau 2 kelompok data yang baru dengan dokumen atau kelompok data lain diambil dari nilai jarak terkecil dari dokumen-dokumen di dalam dari kelompok baru dengan kelompok atau dokumen lainnya. Hasil dari penggabungan dokumen-dokumen tersebut di-*plot* ke dendrogram. Hasil keluaran dari proses ini yaitu *cluster* dokumen secara pertahap.

4.2.5 Penggabungan dengan *Complete Linkage*

Penggabungan 2 dokumen dengan *Complete Linkage* merupakan tahap *clustering* dengan memperhatikan jarak (nilai *Cosine Distance)* terjauh atau terbesar. Langkah-langkah untuk penggabungan dokumen atau *cluster* dengan *complete linkage* dapat dilihat pada Gambar 4.9.


Mulai
Nilai
Cosine Distance
(CosDisDoc)
for doc1 in cluster
for doc2 in cluster
if doc1 not list
and doc2 not list
False
elif doc1 not list
and doc2 is list
False
elif doc1 is list
and doc2 is list
False
True
True
True
listCosDis = Nilai
Cosine Distance
doc1,doc2
listCosDis = Maximum
nilai Cosine Distance
doc1 dengan seluruh
element doc2
listCosDis = Maximum
nilai Cosine Distance
seluruh element doc1
dengan seluruh
element doc2
doc2
doc1
dokDiambil = minimum
nilai dari listCosDis
for object in cluster
if element
dokDiambil in
object
False
True
Gabung 2 object yang
terdapat element
dokDiambil
object
A
B

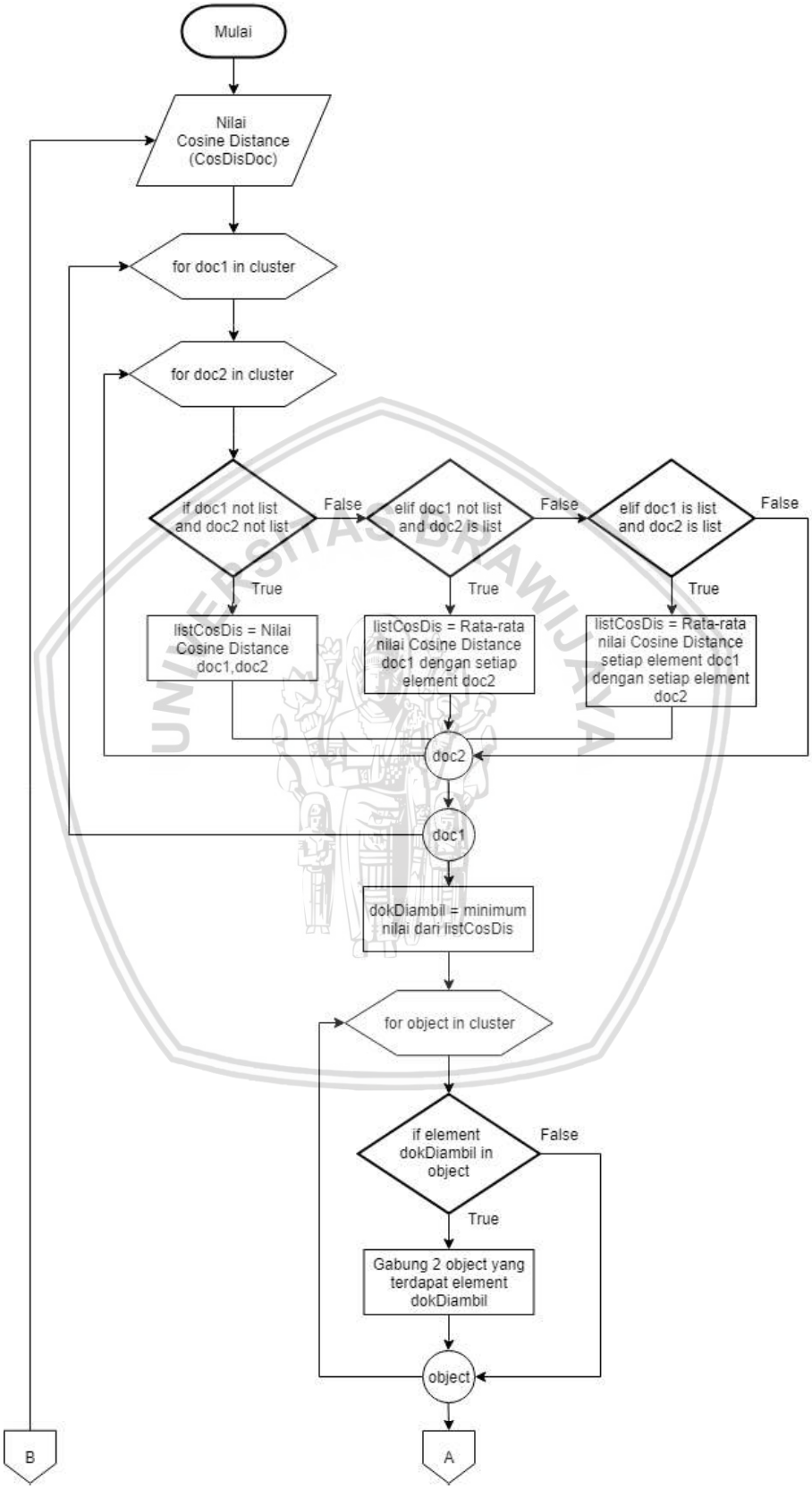


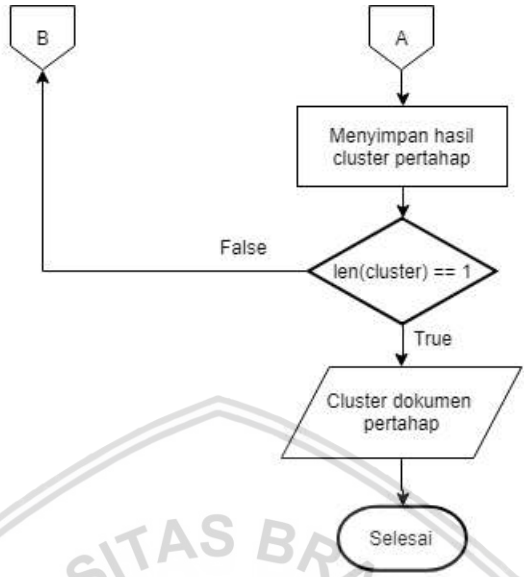
Gambar 4.9 Diagram alir penggabungan data dengan *Complete Linkage*

Masukan pada proses penggabungan data ini yaitu nilai *cosine distance* dari tiap-tiap dokumen dengan dokumen lain. Metode *complete linkage* akan menggabungkan 2 dokumen atau 2 kelompok data dengan nilai *cosine distance* terdekat. Setelah 2 dokumen atau 2 kelompok data digabungkan selanjutnya akan meng-*update* kembali nilai *cosine distance* dengan melibatkan 2 dokumen atau 2 kelompok data. Jika menggunakan *complete linkage* maka nilai *cosine distance* dari 2 dokumen atau 2 kelompok data yang baru dengan dokumen atau kelompok data lain diambil dari nilai jarak terjauh dari dokumen-dokumen di dalam dari kelompok baru dengan kelompok atau dokumen lainnya. Hasil dari penggabungan dokumen-dokumen tersebut disimpan secara pertahap. Hasil keluaran dari proses ini yaitu *cluster* dokumen secara pertahap.

4.2.6 Penggabungan dengan *Average Linkage*

Penggabungan 2 dokumen dengan *Average Linkage* merupakan tahap *clustering* dengan memperhatikan jarak (nilai *Cosine Distance)* rata-rata dari 2 dokumen. Langkah-langkah untuk penggabungan dokumen atau *cluster* dengan *average linkage* dapat dilihat pada Gambar 4.10,


Mulai
Nilai Cosine Distance (CosDisDoc)
for doc1 in cluster
for doc2 in cluster
if doc1 not list and doc2 not list
False
elif doc1 not list and doc2 is list
False
elif doc1 is list and doc2 is list
False
True
True
True
listCosDis = Nilai Cosine Distance doc1,doc2
listCosDis = Rata-rata nilai Cosine Distance doc1 dengan setiap element doc2
listCosDis = Rata-rata nilai Cosine Distance setiap element doc1 dengan setiap element doc2
doc2
doc1
dokDiambil = minimum nilai dari listCosDis
for object in cluster
if element dokDiambil in object
False
True
Gabung 2 object yang terdapat element dokDiambil
object
B
A

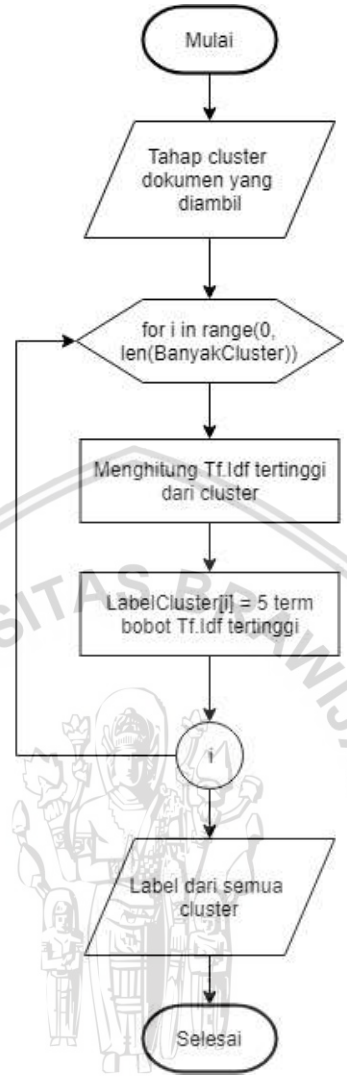


Gambar 4.10 Diagram alir penggabungan data dengan *Average Linkage*

Masukan pada proses penggabungan data ini yaitu nilai *cosine distance* dari tiap-tiap dokumen dengan dokumen lain. Metode *average linkage* akan menggabungkan 2 dokumen atau 2 kelompok data dengan nilai *cosine distance* terdekat dari 2 dokumen. Setelah 2 dokumen atau 2 kelompok data digabungkan selanjutnya akan meng-*update* kembali nilai *cosine distance* dengan melibatkan 2 dokumen atau 2 kelompok data. Jika menggunakan *average linkage* maka nilai *cosine distance* dari 2 dokumen atau 2 kelompok data yang baru dengan dokumen atau kelompok data lain diambil dari nilai jarak rata-rata dari dokumen-dokumen di dalam dari kelompok baru dengan kelompok atau dokumen lainnya. Hasil dari penggabungan dokumen-dokumen tersebut disimpan secara pertahap. Hasil keluaran dari proses ini yaitu *cluster* dokumen secara pertahap.

4.2.7 Memunculkan label *cluster*

Label *cluster* digunakan untuk memberikan identitas pada suatu *cluster* sehingga suatu *cluster* dapat dikenali. Langkah-langkah untuk memunculkan label *cluster* dapat dilihat pada Gambar 4.11.

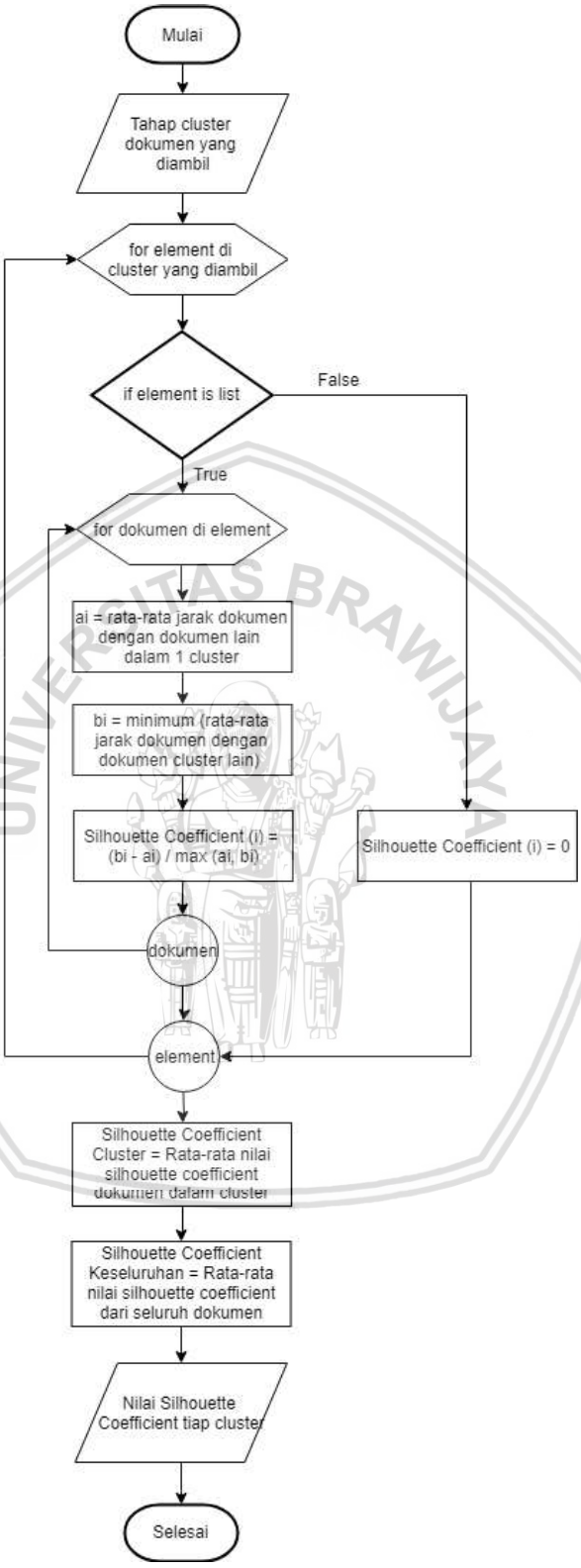


Gambar 4.11 Diagram alir memunculkan label *cluster*

Masukan dari tahap ini yaitu *cluster-cluster* dokumen yang telah terbentuk. Setiap *cluster* dihitung bobot *term* yang muncul pada *cluster* tersebut kemudian diambil 5-10 *term* dengan bobot *term* tertinggi sebagai label dari suatu *cluster.* Hasil keluaran dari tahap ini yaitu *cluster-cluster* yang memiliki label.

4.2.8 Evaluasi

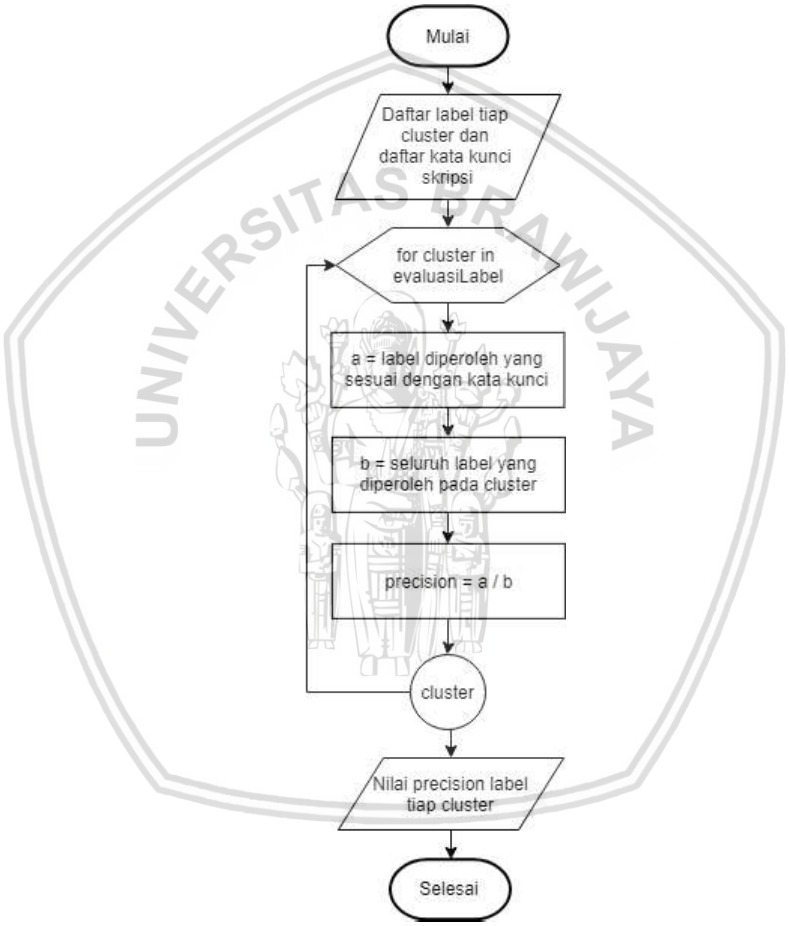
Evaluasi berupa pengujian digunakan untuk mengetahui seberapa baik *cluster-clsuter* yang telah dihasilkan dari metode *Hierarchical Agglomerative Clustering.* Metode pengujian yang digunakan adalah *Silhouette Coefficient.* Langkah-langkah untuk evaluasi dapat dilihat pada Gambar 4.12.



Gambar 4.12 Diagram alir evaluasi *Silhouette Coefficient*

Masukan untuk tahap evaluasi ini yaitu *cluster-cluster* dokumen berlabel yang sudah terbentuk. Tahap pertama yaitu menghitung rata-rata jarak tiap-tiap dokumen dalam satu *cluster* tertentu. Tahap kedua yaitu mengambil nilai minimum dari rata-rata nilai jarak tiap-tiap dokumen dari suatu *cluster* dengan tiap-tiap dokumen dari *cluster* lain. Tahap ketiga yaitu menghitung nilai *Silhouette Coefficient,* untuk detail rumus perhitungan dari *Silhouette Coefficient* dapat dilihat pada Sub-Bab 2.8.1 Persamaan 2.7. Hasil keluaran dari tahap ini yaitu nilai *Silhouette Coefficient* dari masing-masing *cluster.*

Evaluasi hasil pelabelan menggunakan metode *precision.* Langkah-langkah untuk evaluasi hasil pelabelan dengan *precision* dapat dilihat pada gambar 4.13.



Gambar 4.13 Diagram alir evaluasi *Precision*

Masukan dari proses evaluasi label dengan *precision* adalah daftar label tiap *cluster* dan daftar kata kunci dokumen skripsi. Nilai a diperoleh dari jumlah label seluruh dokumen pada *cluster* yang sesuai dengan kata kunci dokumen skripsi pada *cluster* tersebut dan nilai b diperoleh dari jumlah seluruh label yang diperoleh pada *cluster* tersebut. Keluaran dari proses ini adalah nilai *precision* tiap dari label tiap *cluster.*

4.3 Perhitungan Manual

Contoh perhitungan manual yang dilakukan dalam penelitian ini akan menggunakan data sampel sebanyak 5 dokumen dan pada bagian abstrak skripsi hanya diambil 2 kalimat terakhir untuk masing-masing 5 dokumen. Data yang digunakan sebagai data sampel dapat dilihat pada Tabel 4.1. Data pada tabel 4.1 merupakan data judul dan abstrak skripsi yang disimpan dalam format *CSV.*

Tabel 4.1 Sampel data judul dan abstrak skripsi

Dokumen ke-	Judul
0	Analisis dan Implementasi Load Balancing pada Web Server dengan Algoritma Least Connection.
1	Analisis Dan Pemodelan Sistem Pengadaan Barang Dengan Supply Chain Management Pada Pabrik Toys Factory Baiducha Technology
2	Analisis Dan Rancang Bangun Aplikasi Enterprise Resource Planning Education Pada SMKN 2 Buduran.
3	Analisis Implementasi persona Pada Penerapan Metode Evaluasi Usability Heuristic Evaluation. Studi Kasus : Situs Web Fakultas Ilmu Komputer Universitas Brawijaya (Filkom UB)
4	Analisis Kinerja Protokol Routing Open Shortest Path First Pada Teknologi Software-Defined Networking
5	Analisis Openflow Load Balancing Web Server Dengan Algoritma Least Connection Pada Software Defined Network
6	Analisis Perbandingan Algoritma Dijkstra Dan Bellman-Ford Untuk Menentukan Rute Terpendek Dengan Menggunakan Software Defined Network
7	Analisis Protokol routing OSPF (Open Shortest Path First) pada Wireless Mesh Network dalam kasus traffic data Voice
8	Analisis Sentimen Opini Film Berbahasa Indonesia Berbasis Kamus Menggunakan Metode Neighbor- Weigthed K-Nearest Neighbor
9	Analisis Sentimen Pencitraan Elite Politik Berdasarkan Opini Melalui Media Sosial Twitter Menggunakan Metode Additive Kernel SVM

Data judul dan abstrak skripsi tersebut selanjutnya dilakukan proses *preprocessing* sehingga menghasilkan *term-term* penting beserta nilai frekuensi kemunculan pada setiap dokumen. Frekuensi kemunculan *term* pada suatu dokumen dapat dilihat pada tabel 4.2.

Tabel 4.2 Frekuensi *term* tiap dokumen

Term	Doc 0	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8	Doc 9
2	0	0	1	0	0	0	0	0	0	0
additive	0	0	0	0	0	0	0	0	0	1
algoritma	1	0	0	0	0	1	1	0	0	0
analisis	1	1	1	1	1	1	1	1	1	1
aplikasi	0	0	1	0	0	0	0	0	0	0

bahasa	0	0	0	0	0	0	0	0	1	0
baiducha	0	1	0	0	0	0	0	0	0	0
balancing	1	0	0	0	0	1	0	0	0	0
banding	0	0	0	0	0	0	1	0	0	0
bangun	0	0	1	0	0	0	0	0	0	0
barang	0	1	0	0	0	0	0	0	0	0
bas	0	0	0	0	0	0	0	0	1	0
bellman-ford	0	0	0	0	0	0	1	0	0	0
brawijaya	0	0	0	1	0	0	0	0	0	0
budur	0	0	1	0	0	0	0	0	0	0
chain	0	1	0	0	0	0	0	0	0	0
citra	0	0	0	0	0	0	0	0	0	1
connection	1	0	0	0	0	1	0	0	0	0
dasar	0	0	0	0	0	0	0	0	0	1
data	0	0	0	0	0	0	0	1	0	0
defined	0	0	0	0	0	1	1	0	0	0
dijkstra	0	0	0	0	0	0	1	0	0	0
education	0	0	1	0	0	0	0	0	0	0
elite	0	0	0	0	0	0	0	0	0	1
enterprise	0	0	1	0	0	0	0	0	0	0
evaluasi	0	0	0	1	0	0	0	0	0	0
evaluation	0	0	0	1	0	0	0	0	0	0
factory	0	1	0	0	0	0	0	0	0	0
fakultas	0	0	0	1	0	0	0	0	0	0
filkom	0	0	0	1	0	0	0	0	0	0
film	0	0	0	0	0	0	0	0	1	0
first	0	0	0	0	1	0	0	1	0	0
heuristic	0	0	0	1	0	0	0	0	0	0
ilmu	0	0	0	1	0	0	0	0	0	0
implementasi	1	0	0	1	0	0	0	0	0	0
indonesia	0	0	0	0	0	0	0	0	1	0
k-nearest	0	0	0	0	0	0	0	0	1	0
kamus	0	0	0	0	0	0	0	0	1	0
kasus	0	0	0	1	0	0	0	1	0	0
kerja	0	0	0	0	1	0	0	0	0	0
kernel	0	0	0	0	0	0	0	0	0	1
komputer	0	0	0	1	0	0	0	0	0	0
lalu	0	0	0	0	0	0	0	0	0	1
least	1	0	0	0	0	1	0	0	0	0
load	1	0	0	0	0	1	0	0	0	0
management	0	1	0	0	0	0	0	0	0	0
media	0	0	0	0	0	0	0	0	0	1
mesh	0	0	0	0	0	0	0	1	0	0
metode	0	0	0	1	0	0	0	0	1	1
model	0	1	0	0	0	0	0	0	0	0

neighbor	0	0	0	0	0	0	0	0	1	0
neighbor-	0	0	0	0	0	0	0	0	1	0
network	0	0	0	0	0	1	1	1	0	0
networking	0	0	0	0	1	0	0	0	0	0
open	0	0	0	0	1	0	0	1	0	0
openflow	0	0	0	0	0	1	0	0	0	0
opini	0	0	0	0	0	0	0	0	1	1
ospf	0	0	0	0	0	0	0	1	0	0
pabrik	0	1	0	0	0	0	0	0	0	0
path	0	0	0	0	1	0	0	1	0	0
pendek	0	0	0	0	0	0	1	0	0	0
persona	0	0	0	1	0	0	0	0	0	0
planning	0	0	1	0	0	0	0	0	0	0
politik	0	0	0	0	0	0	0	0	0	1
protokol	0	0	0	0	1	0	0	1	0	0
rancang	0	0	1	0	0	0	0	0	0	0
resource	0	0	1	0	0	0	0	0	0	0
routing	0	0	0	0	1	0	0	1	0	0
rute	0	0	0	0	0	0	1	0	0	0
sentimen	0	0	0	0	0	0	0	0	1	1
server	1	0	0	0	0	1	0	0	0	0
shortest	0	0	0	0	1	0	0	1	0	0
sistem	0	1	0	0	0	0	0	0	0	0
situs	0	0	0	1	0	0	0	0	0	0
smkn	0	0	1	0	0	0	0	0	0	0
software	0	0	0	0	0	1	1	0	0	0
software-defined	0	0	0	0	1	0	0	0	0	0
sosial	0	0	0	0	0	0	0	0	0	1
studi	0	0	0	1	0	0	0	0	0	0
supply	0	1	0	0	0	0	0	0	0	0
svm	0	0	0	0	0	0	0	0	0	1
technology	0	1	0	0	0	0	0	0	0	0
teknologi	0	0	0	0	1	0	0	0	0	0
terap	0	0	0	1	0	0	0	0	0	0
toys	0	1	0	0	0	0	0	0	0	0
traffic	0	0	0	0	0	0	0	1	0	0
twitter	0	0	0	0	0	0	0	0	0	1
Ub	0	0	0	1	0	0	0	0	0	0
universitas	0	0	0	1	0	0	0	0	0	0
usability	0	0	0	1	0	0	0	0	0	0
voice	0	0	0	0	0	0	0	1	0	0
web	1	0	0	1	0	1	0	0	0	0
weigthed	0	0	0	0	0	0	0	0	1	0
wireless	0	0	0	0	0	0	0	1	0	0

4.3.1 Perhitungan Manual Pembobotan *TF-IDF*

Tahap pertama untuk menghitung pembobotan *TF-IDF* yaitu menghitung terlebih dahulu bobot frekuensi *term (Tf).* Rumus perhitungan pembobotan *Tf* dapat dilihat pada Sub-Bab 2.4 Persamaan 2.1. Tabel 4.3 menunjukan hasil dari perhitungan bobot *Tf.*

Tabel 4.3 Nilai bobot *Tf* pada tiap dokumen

Term	Doc 0	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8	Doc 9
2	0	0	1	0	0	0	0	0	0	0
additive	0	0	0	0	0	0	0	0	0	1
algoritma	1	0	0	0	0	1	1	0	0	0
analisis	1	1	1	1	1	1	1	1	1	1
aplikasi	0	0	1	0	0	0	0	0	0	0
bahasa	0	0	0	0	0	0	0	0	1	0
baiducha	0	1	0	0	0	0	0	0	0	0
balancing	1	0	0	0	0	1	0	0	0	0
banding	0	0	0	0	0	0	1	0	0	0
bangun	0	0	1	0	0	0	0	0	0	0
barang	0	1	0	0	0	0	0	0	0	0
Bas	0	0	0	0	0	0	0	0	1	0
bellman-ford	0	0	0	0	0	0	1	0	0	0
brawijaya	0	0	0	1	0	0	0	0	0	0
Budur	0	0	1	0	0	0	0	0	0	0
Chain	0	1	0	0	0	0	0	0	0	0
Citra	0	0	0	0	0	0	0	0	0	1
connection	1	0	0	0	0	1	0	0	0	0
Dasar	0	0	0	0	0	0	0	0	0	1
Data	0	0	0	0	0	0	0	1	0	0
defined	0	0	0	0	0	1	1	0	0	0
dijkstra	0	0	0	0	0	0	1	0	0	0
education	0	0	1	0	0	0	0	0	0	0
Elite	0	0	0	0	0	0	0	0	0	1
enterprise	0	0	1	0	0	0	0	0	0	0
evaluasi	0	0	0	1	0	0	0	0	0	0
evaluation	0	0	0	1	0	0	0	0	0	0
factory	0	1	0	0	0	0	0	0	0	0
fakultas	0	0	0	1	0	0	0	0	0	0
Filkom	0	0	0	1	0	0	0	0	0	0
Film	0	0	0	0	0	0	0	0	1	0
First	0	0	0	0	1	0	0	1	0	0
heuristic	0	0	0	1	0	0	0	0	0	0
Ilmu	0	0	0	1	0	0	0	0	0	0
implementasi	1	0	0	1	0	0	0	0	0	0
indonesia	0	0	0	0	0	0	0	0	1	0

k-nearest	0	0	0	0	0	0	0	0	1	0
kamus	0	0	0	0	0	0	0	0	1	0
Kasus	0	0	0	1	0	0	0	1	0	0
Kerja	0	0	0	0	1	0	0	0	0	0
kernel	0	0	0	0	0	0	0	0	0	1
komputer	0	0	0	1	0	0	0	0	0	0
Lalu	0	0	0	0	0	0	0	0	0	1
Least	1	0	0	0	0	1	0	0	0	0
Load	1	0	0	0	0	1	0	0	0	0
management	0	1	0	0	0	0	0	0	0	0
Media	0	0	0	0	0	0	0	0	0	1
Mesh	0	0	0	0	0	0	0	1	0	0
metode	0	0	0	1	0	0	0	0	1	1
model	0	1	0	0	0	0	0	0	0	0
neighbor	0	0	0	0	0	0	0	0	1	0
neighbor-	0	0	0	0	0	0	0	0	1	0
network	0	0	0	0	0	1	1	1	0	0
networking	0	0	0	0	1	0	0	0	0	0
open	0	0	0	0	1	0	0	1	0	0
openflow	0	0	0	0	0	1	0	0	0	0
opini	0	0	0	0	0	0	0	0	1	1
ospf	0	0	0	0	0	0	0	1	0	0
pabrik	0	1	0	0	0	0	0	0	0	0
path	0	0	0	0	1	0	0	1	0	0
pendek	0	0	0	0	0	0	1	0	0	0
persona	0	0	0	1	0	0	0	0	0	0
planning	0	0	1	0	0	0	0	0	0	0
politik	0	0	0	0	0	0	0	0	0	1
protokol	0	0	0	0	1	0	0	1	0	0
rancang	0	0	1	0	0	0	0	0	0	0
resource	0	0	1	0	0	0	0	0	0	0
routing	0	0	0	0	1	0	0	1	0	0
rute	0	0	0	0	0	0	1	0	0	0
sentimen	0	0	0	0	0	0	0	0	1	1
server	1	0	0	0	0	1	0	0	0	0
shortest	0	0	0	0	1	0	0	1	0	0
sistem	0	1	0	0	0	0	0	0	0	0
situs	0	0	0	1	0	0	0	0	0	0
smkn	0	0	1	0	0	0	0	0	0	0
software	0	0	0	0	0	1	1	0	0	0
software-defined	0	0	0	0	1	0	0	0	0	0
sosial	0	0	0	0	0	0	0	0	0	1
studi	0	0	0	1	0	0	0	0	0	0
supply	0	1	0	0	0	0	0	0	0	0

svm	0	0	0	0	0	0	0	0	0	1
technology	0	1	0	0	0	0	0	0	0	0
teknologi	0	0	0	0	1	0	0	0	0	0
terap	0	0	0	1	0	0	0	0	0	0
toys	0	1	0	0	0	0	0	0	0	0
traffic	0	0	0	0	0	0	0	1	0	0
twitter	0	0	0	0	0	0	0	0	0	1
Ub	0	0	0	1	0	0	0	0	0	0
universitas	0	0	0	1	0	0	0	0	0	0
usability	0	0	0	1	0	0	0	0	0	0
voice	0	0	0	0	0	0	0	1	0	0
web	1	0	0	1	0	1	0	0	0	0
weigthed	0	0	0	0	0	0	0	0	1	0
wireless	0	0	0	0	0	0	0	1	0	0

Tahap selanjutnya yaitu menghitung nilai *inverse document frequency (idf)* yaitu nilai *inverse* dari jumlah dokumen di mana suatu *term* muncul *(df).* Rumus perhitungan *idf* dapat dilihat pada Sub-Bab 2.4 Persamaan 2.2. Tabel 4.4 menunjukkan hasil perhitungan manual nilai *idf* tiap-tiap *term.*

Tabel 4.4 Nilai *idf* tiap-tiap *term*

Term	DF	idf
2	1	1
additive	1	1
algoritma	3	0,522879
analisis	10	0
aplikasi	1	1
bahasa	1	1
baiducha	1	1
balancing	2	0,69897
banding	1	1
bangun	1	1
barang	1	1
bas	1	1
bellman-ford	1	1
brawijaya	1	1
budur	1	1
chain	1	1
citra	1	1
connection	2	0,69897
dasar	1	1
data	1	1
defined	2	0,69897
dijkstra	1	1

education	1	1
elite	1	1
enterprise	1	1
evaluasi	1	1
evaluation	1	1
factory	1	1
fakultas	1	1
filkom	1	1
film	1	1
first	2	0,69897
heuristic	1	1
ilmu	1	1
implementasi	2	0,69897
indonesia	1	1
k-nearest	1	1
kamus	1	1
kasus	2	0,69897
kerja	1	1
kernel	1	1
komputer	1	1
lalu	1	1
least	2	0,69897
load	2	0,69897
management	1	1
media	1	1
mesh	1	1
metode	3	0,522879
model	1	1
neighbor	1	1
neighbor-	1	1
network	3	0,522879
networking	1	1
open	2	0,69897
openflow	1	1
opini	2	0,69897
ospf	1	1
pabrik	1	1
path	2	0,69897
pendek	1	1
persona	1	1
planning	1	1
politik	1	1
protokol	2	0,69897
rancang	1	1
resource	1	1

routing	2	0,69897
rute	1	1
sentimen	2	0,69897
server	2	0,69897
shortest	2	0,69897
sistem	1	1
situs	1	1
smkn	1	1
software	2	0,69897
software-defined	1	1
sosial	1	1
studi	1	1
supply	1	1
svm	1	1
technology	1	1
teknologi	1	1
terap	1	1
toys	1	1
traffic	1	1
twitter	1	1
ub	1	1
universitas	1	1
usability	1	1
voice	1	1
web	3	0,522879
weigthed	1	1
wireless	1	1

Tahap selanjutnya yaitu mengalikan nilai *idf* dengan masing-masing nilai *Tf*. Detail rumus perhitungan dapat dilihat pada Sub-Bab 2.4 Persamaan 2.3. Tabel 4.5 menunjukan hasil dari pembobotan *TF-IDF*.

Tabel 4.5 Hasil pembobotan *TF-IDF*

Term	**Doc0**	**Doc1**	**Doc2**	**Doc3**	**Doc4**	**Doc5**	**Doc6**	**Doc7**	**Doc8**	**Doc9**
2	0	0	1	0	0	0	0	0	0	0
additive	0	0	0	0	0	0	0	0	0	1
algoritma	0,523	0	0	0	0	0,523	0,523	0	0	0
Analisis	0	0	0	0	0	0	0	0	0	0
aplikasi	0	0	1	0	0	0	0	0	0	0
Bahasa	0	0	0	0	0	0	0	0	1	0
baiducha	0	1	0	0	0	0	0	0	0	0
balancing	0,699	0	0	0	0	0,699	0	0	0	0
banding	0	0	0	0	0	0	1	0	0	0
Bangun	0	0	1	0	0	0	0	0	0	0

Barang	0	1	0	0	0	0	0	0	0	0
Bas	0	0	0	0	0	0	0	0	1	0
bellman-ford	0	0	0	0	0	0	1	0	0	0
brawijaya	0	0	0	1	0	0	0	0	0	0
Budur	0	0	1	0	0	0	0	0	0	0
Chain	0	1	0	0	0	0	0	0	0	0
Citra	0	0	0	0	0	0	0	0	0	1
connection	0,699	0	0	0	0	0,699	0	0	0	0
Dasar	0	0	0	0	0	0	0	0	0	1
Data	0	0	0	0	0	0	0	1	0	0
defined	0	0	0	0	0	0,699	0,699	0	0	0
Dijkstra	0	0	0	0	0	0	1	0	0	0
education	0	0	1	0	0	0	0	0	0	0
Elite	0	0	0	0	0	0	0	0	0	1
enterprise	0	0	1	0	0	0	0	0	0	0
evaluasi	0	0	0	1	0	0	0	0	0	0
evaluation	0	0	0	1	0	0	0	0	0	0
Factory	0	1	0	0	0	0	0	0	0	0
fakultas	0	0	0	1	0	0	0	0	0	0
Filkom	0	0	0	1	0	0	0	0	0	0
Film	0	0	0	0	0	0	0	0	1	0
First	0	0	0	0	0,699	0	0	0,699	0	0
heuristic	0	0	0	1	0	0	0	0	0	0
Ilmu	0	0	0	1	0	0	0	0	0	0
implementasi	0,699	0	0	0,699	0	0	0	0	0	0
indonesia	0	0	0	0	0	0	0	0	1	0
k-nearest	0	0	0	0	0	0	0	0	1	0
Kamus	0	0	0	0	0	0	0	0	1	0
Kasus	0	0	0	0,699	0	0	0	0,699	0	0
Kerja	0	0	0	0	1	0	0	0	0	0
Kernel	0	0	0	0	0	0	0	0	0	1
komputer	0	0	0	1	0	0	0	0	0	0
Lalu	0	0	0	0	0	0	0	0	0	1
Least	0,699	0	0	0	0	0,699	0	0	0	0
Load	0,699	0	0	0	0	0,699	0	0	0	0
management	0	1	0	0	0	0	0	0	0	0
Media	0	0	0	0	0	0	0	0	0	1
Mesh	0	0	0	0	0	0	0	1	0	0
metode	0	0	0	0,523	0	0	0	0	0,523	0,523
Model	0	1	0	0	0	0	0	0	0	0
neighbor	0	0	0	0	0	0	0	0	1	0
neighbor-	0	0	0	0	0	0	0	0	1	0
network	0	0	0	0	0	0,523	0,523	0,523	0	0
networking	0	0	0	0	1	0	0	0	0	0
Open	0	0	0	0	0,699	0	0	0,699	0	0

openflow	0	0	0	0	0	1	0	0	0	0
Opini	0	0	0	0	0	0	0	0	0,699	0,699
Ospf	0	0	0	0	0	0	0	1	0	0
Pabrik	0	1	0	0	0	0	0	0	0	0
Path	0	0	0	0	0,699	0	0	0,699	0	0
Pendek	0	0	0	0	0	0	1	0	0	0
persona	0	0	0	1	0	0	0	0	0	0
planning	0	0	1	0	0	0	0	0	0	0
Politik	0	0	0	0	0	0	0	0	0	1
protokol	0	0	0	0	0,699	0	0	0,699	0	0
rancang	0	0	1	0	0	0	0	0	0	0
resource	0	0	1	0	0	0	0	0	0	0
Routing	0	0	0	0	0,699	0	0	0,699	0	0
Rute	0	0	0	0	0	0	1	0	0	0
sentimen	0	0	0	0	0	0	0	0	0,699	0,699
Server	0,699	0	0	0	0	0,699	0	0	0	0
shortest	0	0	0	0	0,699	0	0	0,699	0	0
System	0	1	0	0	0	0	0	0	0	0
Situs	0	0	0	1	0	0	0	0	0	0
Smkn	0	0	1	0	0	0	0	0	0	0
software	0	0	0	0	0	0,699	0,699	0	0	0
software-defined	0	0	0	0	1	0	0	0	0	0
Social	0	0	0	0	0	0	0	0	0	1
Studi	0	0	0	1	0	0	0	0	0	0
Supply	0	1	0	0	0	0	0	0	0	0
Svm	0	0	0	0	0	0	0	0	0	1
technology	0	1	0	0	0	0	0	0	0	0
teknologi	0	0	0	0	1	0	0	0	0	0
Terap	0	0	0	1	0	0	0	0	0	0
Toys	0	1	0	0	0	0	0	0	0	0
Traffic	0	0	0	0	0	0	0	1	0	0
Twitter	0	0	0	0	0	0	0	0	0	1
Ub	0	0	0	1	0	0	0	0	0	0
universitas	0	0	0	1	0	0	0	0	0	0
usability	0	0	0	1	0	0	0	0	0	0
Voice	0	0	0	0	0	0	0	1	0	0
Web	0,523	0	0	0,523	0	0,523	0	0	0	0
weigthed	0	0	0	0	0	0	0	0	1	0
wireless	0	0	0	0	0	0	0	1	0	0

4.3.2 Perhitungan Manual *Cosine Distance*

Tahap awal perhitungan manual pada *cosine distance* yaitu dengan memasukkan nilai bobot *TF-IDF* ke perhitungan nilai *cosine similarity.* Rumus perhitungan *cosine similarity* dapat dilihat pada Sub-Bab 2.5 Persamaan 2.4. Nilai

cosine distance diperoleh dari hasil 1 dikurangi nilai *cosine similarity* antar 2 dokumen. Rumus perhitungan *cosine distance* dapat dilihat pada Sub-Bab 2.5.1 Persamaan 2.5. Tabel 4.6 menunjukan hasil perhitungan manual *cosine distance.*

Tabel 4.6 Hasil perhitungan *Cosine Distance*

CosDis	Doc0	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6	Doc7	Doc8	Doc9
Doc0	0	1	1	0,899	1	0,3	0,943	1	1	1
Doc1	1	0	1	1	1	1	1	1	1	1
Doc2	1	1	0	1	1	1	1	1	1	1
Doc3	0,899	1	1	0	1	0,971	1	0,961	0,979	0,981
Doc4	1	1	1	1	0	1	1	0,642	1	1
Doc5	0,3	1	1	0,971	1	0	0,739	0,962	1	1
Doc6	0,943	1	1	1	1	0,739	0	0,966	1	1
Doc7	1	1	1	0,961	0,642	0,962	0,966	0	1	1
Doc8	1	1	1	0,979	1	1	1	1	0	0,888
Doc9	1	1	1	0,981	1	1	1	1	0,888	0

4.3.3 Tahap *Clustering*

Tahap *clustering* yang dilakukan yaitu menggunakan pengelompokan secara berulang mulai dari dokumen-dokumen bersifat individu menuju ke 1 *cluster* besar sehingga membentuk *hierarchical cluster.* Pengelompokan dokumen menggunakan teknik *single linkage* yaitu memperhatikan jarak terdekat dokumen. Setiap setelah 2 dokumen atau 2 kelompok digabungkan, nilai *cosine distance* diperbarui dengan melihat jarak terdekat dari dokumen-dokumen yang ada di dalam kelompok dengan dokumen dari kelompok lain.

Tabel 4.7 menunjukan penggabungan tahap 1 yaitu penggabungan Doc0 dengan Doc5 karena 2 dokumen tersebut memiliki jarak terdekat dibanding dengan dokumen lain.

Tabel 4.7 Penggabungan tahap 1

CosDis	Doc1	Doc2	Doc3	Doc4	Doc6	Doc7	Doc8	Doc9	(Doc0, Doc5)
Doc1	0	1	1	1	1	1	1	1	1
Doc2	1	0	1	1	1	1	1	1	1
Doc3	1	1	0	1	1	0,961	0,979	0,981	0,899
Doc4	1	1	1	0	1	0,642	1	1	1
Doc6	1	1	1	1	0	0,966	1	1	0,739
Doc7	1	1	0,961	0,642	0,966	0	1	1	0,962
Doc8	1	1	0,979	1	1	1	0	0,888	1
Doc9	1	1	0,981	1	1	1	0,888	0	1
(Doc0, Doc5)	1	1	0,899	1	0,739	0,962	1	1	0

Tahap berikutnya yaitu meng-*update* nilai *Cosine Distance* kemudian kembali menggabungkan dokumen. Tabel 4.8 menunjukan penggabungan tahap 2 yaitu penggabungan Doc4 dengan Doc7.

Tabel 4.8 Penggabungan tahap 2

CosDis	Doc1	Doc2	Doc3	Doc6	Doc8	Doc9	(Doc0, Doc5)	(Doc4, Doc7)
Doc1	0	1	1	1	1	1	1	1
Doc2	1	0	1	1	1	1	1	1
Doc3	1	1	0	1	0,979	0,981	0,899	0,961
Doc6	1	1	1	0	1	1	0,739	0,966
Doc8	1	1	0,979	1	0	0,888	1	1
Doc9	1	1	0,981	1	0,888	0	1	1
(Doc0, Doc5)	1	1	0,899	0,739	1	1	0	0,962
(Doc4, Doc7)	1	1	0,961	0,966	1	1	0,962	0

Nilai dari *cosine distance* kembali diperbarui kemudian dilanjutkan penggabungan dokumen kembali. Tabel 4.9 menunjukan penggabungan tahap 3 yaitu penggabungan Doc6 dengan *cluster* (Doc0, Doc5).

Tabel 4.9 Penggabungan tahap 3

CosDis	Doc1	Doc2	Doc3	Doc8	Doc9	(Doc4,Doc7)	(Doc0,Doc5,Doc6)
Doc1	0	1	1	1	1	1	1
Doc2	1	0	1	1	1	1	1
Doc3	1	1	0	0,979	0,981	0,961	0,899
Doc8	1	1	0,979	0	0,888	1	1
Doc9	1	1	0,981	0,888	0	1	1
(Doc4,Doc7)	1	1	0,961	1	1	0	0,962
(Doc0,Doc5,Doc6)	1	1	0,899	1	1	0,962	0

Tabel 4.10 menunjukan penggabungan tahap 4 yaitu penggabungan antara Doc8 dan Doc9.

Tabel 4.10 Penggabungan tahap 4

CosDis	Doc1	Doc2	Doc3	(Doc4,Doc7)	(Doc0,Doc5,Doc6)	(Doc8,Doc9)
Doc1	0	1	1	1	1	1
Doc2	1	0	1	1	1	1
Doc3	1	1	0	0,961	0,899	0,979
(Doc4,Doc7)	1	1	0,961	0	0,962	1

(Doc0,Doc5,Doc6)	1	1	0,899	0,739	0	1
(Doc8,Doc9)	1	1	0,979	1	1	0

Tabel 4.11 menunjukan penggabungan tahap 5 yaitu penggabungan antara Doc3 dengan *cluster* (Doc0,Doc5,Doc6).

Tabel 4.11 Penggabungan tahap 5

CosDis	**Doc1**	**Doc2**	**(Doc4,Doc7)**	**(Doc8,Doc9)**	**(Doc0,Doc5,Doc6,Doc3)**
Doc1	0	1	1	1	1
Doc2	1	0	1	1	1
(Doc4,Doc7)	1	1	0	1	0,961
(Doc8,Doc9)	1	1	1	0	0,979
(Doc0,Doc5,Doc6,Doc3)	1	1	0,961	0,979	0

Tabel 4.12 menunjukan penggabungan tahap 6 yaitu penggabungan antara *cluster* (Doc4,Doc7) dengan *cluster* (Doc0,Doc5,Doc6,Doc3).

Tabel 4.12 Penggabungan tahap 6

CosDis	**Doc1**	**Doc2**	**(Doc8,Doc9)**	**(Doc4,Doc7,Doc0,Doc5,Doc6,Doc3)**
Doc1	0	1	1	1
Doc2	1	0	1	1
(Doc8,Doc9)	1	1	0	0,979
(Doc4,Doc7,Doc0,Doc5,Doc6,Doc3)	1	1	0,979	0

Tabel 4.13 menunjukan penggabungan tahap 7 yaitu penggabungan antara *cluster* (Doc4,Doc7) dengan *cluster* (Doc0,Doc5,Doc6,Doc3).

Tabel 4.13 Penggabungan tahap 7

CosDis	**Doc1**	**Doc2**	**(Doc8,Doc9,Doc4,Doc7,Doc0,Doc5,Doc6,Doc3)**
Doc1	0	1	1
Doc2	1	0	1
(Doc8,Doc9,Doc4,Doc7,Doc0,Doc5,Doc6,Doc3)	1	1	0

Tabel 4.14 menunjukan penggabungan tahap 8 yaitu penggabungan antara Doc1 dengan *cluster* (Doc8,Doc9,Doc4,Doc7, Doc0,Doc5,Doc6,Doc3).

Tabel 4.14 Penggabungan tahap 8

CosDis	Doc2	(Doc1,Doc8,Doc9,Doc4,Doc7,Doc0,Doc5,Doc6,Doc3)
Doc2	0	1
(Doc1,Doc8,Doc9,Doc4,Doc7,Doc0,Doc5,Doc6,Doc3)	1	0

Tabel 4.15 menunjukan penggabungan tahap 9 yaitu penggabungan antara Doc2 dengan *cluster* (Doc1, Doc8,Doc9,Doc4,Doc7, Doc0,Doc5,Doc6,Doc3) yang merupakan tahap penggabungan data terakhir.

Tabel 4.15 Penggabungan tahap 9

CosDis	(Doc0,Doc1,Doc2,Doc3,Doc4,Doc5,Doc6,Doc7,Doc8,Doc9)
(Doc0,Doc1,Doc2,Doc3,Doc4,Doc5,Doc6,Doc7,Doc8,Doc9)	0

Cara untuk memperjelas *cluster-cluster* yang terbentuk dari *hierarchical clustering* dapat menggunakan dendrogram atau bisa dengan menunjukkan setiap tahapan proses terbentuknya *cluster*. Pada penelitian ini menggunakan cara dengan menyimpan setiap tahapan proses terbentuknya *cluster.* Tahap proses pembentukkan *cluster* digunakan sebagai titik potong dalam mengambil *cluster* yang akan dipakai untuk dievaluasi. Tabel 4.16 menunjukkan setiap tahap penggabungan data yang disimpan disimpan untuk mengetahui proses *clustering* yang terbentuk.

Tabel 4.16 Tahap proses pembentukkan *cluster*

Tahap	*Cluster* Dokumen
1	Doc1, Doc2, Doc3, Doc4, Doc6, Doc7, Doc8, Doc9, [Doc0, Doc5]
2	Doc1, Doc2, Doc3, Doc6, Doc8, Doc9, [Doc0, Doc5], [Doc4, Doc7]
3	Doc1, Doc2, Doc3, Doc8, Doc9, [Doc0, Doc5, Doc6], [Doc4, Doc7]
4	Doc1, Doc2, Doc3, [Doc0, Doc5, Doc6], [Doc4, Doc7], [Doc8, Doc9]
5	Doc1, Doc2, [Doc0, Doc5, Doc6, Doc3], [Doc4, Doc7], [Doc8, Doc9]
6	Doc1, Doc2, [Doc8, Doc9], [Doc0, Doc5, Doc6, Doc3, Doc4, Doc7]
7	Doc1, Doc2, [Doc0, Doc5, Doc6, Doc3, Doc4, Doc7, Doc8, Doc9]
8	Doc2, [Doc0, Doc5, Doc6, Doc3, Doc4, Doc7, Doc8, Doc9, Doc1]

9	[Doc0, Doc5, Doc6, Doc3, Doc4, Doc7, Doc8, Doc9, Doc1, Doc2]

4.3.4 Pelabelan *Cluster*

Proses pelabelan *cluster* menggunakan pembobotan *TF-IDF*. Tahapan yang dilalui untuk melakukan pelabelan *cluster* menggunakan pembobotan *TF-IDF* hampir mirip dengan pembobotan *TF-IDF* pada dokumen. Pembobotan *TF-IDF* yang digunakan untuk pelabelan *cluster* yaitu pembobotan yang didasarkan *term* yang muncul pada suatu *cluster.*

Pada manualisasi ini mengambil contoh jika memilih titik potong pada tahap 5 yaitu menghasilkan 5 *cluster.* Tabel 4.17 menunjukkan *cluster* yang dihasilkan pada titik potong tahap 5 beserta dokumen didalam *cluster*.

Tabel 4.17 *Cluster* pada titik potong tahap 5

Cluster	**Dokumen**
C0	Doc1
C1	Doc2
C2	(Doc4,Doc7)
C3	(Doc8,Doc9)
C4	(Doc0,Doc5,Doc6,Doc3)

Tahap pertama yaitu menghitung frekuensi kemunculan *term* pada *cluster.* Tabel 4.18 menunjukan frekuensi *term* yang muncul pada masing-masing *cluster.*

Tabel 4.18 Frekuensi *term* pada *cluster*

Term	**C0**	**C1**	**C2**	**C3**	**C4**
2	0	1	0	0	0
additive	0	0	0	1	0
algoritma	0	0	0	0	3
analisis	1	1	2	2	4
aplikasi	0	1	0	0	0
bahasa	0	0	0	1	0
baiducha	1	0	0	0	0
balancing	0	0	0	0	2
banding	0	0	0	0	1
bangun	0	1	0	0	0
barang	1	0	0	0	0
bas	0	0	0	1	0
bellman-ford	0	0	0	0	1
brawijaya	0	0	0	0	1
budur	0	1	0	0	0
chain	1	0	0	0	0
citra	0	0	0	1	0

connection	0	0	0	0	2
dasar	0	0	0	1	0
data	0	0	1	0	0
defined	0	0	0	0	2
dijkstra	0	0	0	0	1
education	0	1	0	0	0
elite	0	0	0	1	0
enterprise	0	1	0	0	0
evaluasi	0	0	0	0	1
evaluation	0	0	0	0	1
factory	1	0	0	0	0
fakultas	0	0	0	0	1
filkom	0	0	0	0	1
film	0	0	0	1	0
first	0	0	2	0	0
heuristic	0	0	0	0	1
ilmu	0	0	0	0	1
implementasi	0	0	0	0	2
indonesia	0	0	0	1	0
k-nearest	0	0	0	1	0
kamus	0	0	0	1	0
kasus	0	0	1	0	1
kerja	0	0	1	0	0
kernel	0	0	0	1	0
komputer	0	0	0	0	1
lalu	0	0	0	1	0
least	0	0	0	0	2
load	0	0	0	0	2
management	1	0	0	0	0
media	0	0	0	1	0
mesh	0	0	1	0	0
metode	0	0	0	2	1
model	1	0	0	0	0
neighbor	0	0	0	1	0
neighbor-	0	0	0	1	0
network	0	0	1	0	2
networking	0	0	1	0	0
open	0	0	2	0	0
openflow	0	0	0	0	1
opini	0	0	0	2	0
ospf	0	0	1	0	0
pabrik	1	0	0	0	0
path	0	0	2	0	0
pendek	0	0	0	0	1
persona	0	0	0	0	1

planning	0	1	0	0	0
politik	0	0	0	1	0
protokol	0	0	2	0	0
rancang	0	1	0	0	0
resource	0	1	0	0	0
routing	0	0	2	0	0
rute	0	0	0	0	1
sentimen	0	0	0	2	0
server	0	0	0	0	2
shortest	0	0	2	0	0
sistem	1	0	0	0	0
situs	0	0	0	0	1
smkn	0	1	0	0	0
software	0	0	0	0	2
software-defined	0	0	1	0	0
sosial	0	0	0	1	0
studi	0	0	0	0	1
supply	1	0	0	0	0
svm	0	0	0	1	0
technology	1	0	0	0	0
teknologi	0	0	1	0	0
terap	0	0	0	0	1
toys	1	0	0	0	0
traffic	0	0	1	0	0
twitter	0	0	0	1	0
ub	0	0	0	0	1
universitas	0	0	0	0	1
usability	0	0	0	0	1
voice	0	0	1	0	0
web	0	0	0	0	3
weigthed	0	0	0	1	0
wireless	0	0	1	0	0

Tahap berikutnya yaitu menghitung nilai bobot *Tf.* Tabel 4.19 menunjukan nilai bobot *tf* dari masing-masing *cluster.*

Tabel 4.19 Nilai bobot *Tf* pada *cluster*

Term	**C0**	**C1**	**C2**	**C3**	**C4**
2	0	1	0	0	0
additive	0	0	0	1	0
algoritma	0	0	0	0	1,477121
analisis	1	1	1,30103	1,30103	1,60206

aplikasi	0	1	0	0	0
bahasa	0	0	0	1	0
baiducha	1	0	0	0	0
balancing	0	0	0	0	1,30103
banding	0	0	0	0	1
bangun	0	1	0	0	0
barang	1	0	0	0	0
bas	0	0	0	1	0
bellman-ford	0	0	0	0	1
brawijaya	0	0	0	0	1
budur	0	1	0	0	0
chain	1	0	0	0	0
citra	0	0	0	1	0
connection	0	0	0	0	1,30103
dasar	0	0	0	1	0
data	0	0	1	0	0
defined	0	0	0	0	1,30103
dijkstra	0	0	0	0	1
education	0	1	0	0	0
elite	0	0	0	1	0
enterprise	0	1	0	0	0
evaluasi	0	0	0	0	1
evaluation	0	0	0	0	1
factory	1	0	0	0	0
fakultas	0	0	0	0	1
filkom	0	0	0	0	1
film	0	0	0	1	0
first	0	0	1,30103	0	0
heuristic	0	0	0	0	1
ilmu	0	0	0	0	1
implementasi	0	0	0	0	1,30103
indonesia	0	0	0	1	0
k-nearest	0	0	0	1	0
kamus	0	0	0	1	0
kasus	0	0	1	0	1
kerja	0	0	1	0	0
kernel	0	0	0	1	0
komputer	0	0	0	0	1
lalu	0	0	0	1	0
least	0	0	0	0	1,30103
load	0	0	0	0	1,30103
management	1	0	0	0	0
media	0	0	0	1	0
mesh	0	0	1	0	0
metode	0	0	0	1,30103	1

model	1	0	0	0	0
neighbor	0	0	0	1	0
neighbor-	0	0	0	1	0
network	0	0	1	0	1,30103
networking	0	0	1	0	0
open	0	0	1,30103	0	0
openflow	0	0	0	0	1
opini	0	0	0	1,30103	0
ospf	0	0	1	0	0
pabrik	1	0	0	0	0
path	0	0	1,30103	0	0
pendek	0	0	0	0	1
persona	0	0	0	0	1
planning	0	1	0	0	0
politik	0	0	0	1	0
protokol	0	0	1,30103	0	0
rancang	0	1	0	0	0
resource	0	1	0	0	0
routing	0	0	1,30103	0	0
rute	0	0	0	0	1
sentimen	0	0	0	1,30103	0
server	0	0	0	0	1,30103
shortest	0	0	1,30103	0	0
sistem	1	0	0	0	0
situs	0	0	0	0	1
smkn	0	1	0	0	0
software	0	0	0	0	1,30103
software-defined	0	0	1	0	0
sosial	0	0	0	1	0
studi	0	0	0	0	1
supply	1	0	0	0	0
svm	0	0	0	1	0
technology	1	0	0	0	0
teknologi	0	0	1	0	0
terap	0	0	0	0	1
toys	1	0	0	0	0
traffic	0	0	1	0	0
twitter	0	0	0	1	0
ub	0	0	0	0	1
universitas	0	0	0	0	1
usability	0	0	0	0	1
voice	0	0	1	0	0
web	0	0	0	0	1,477121
weigthed	0	0	0	1	0

wireless	0	0	1	0	0

Tahap selanjutnya yaitu menghitung nilai *idf*. Tabel 4.20 menunjukan nilai *idf* dari seluruh *term*.

Tabel 4.20 Nilai *idf* seluruh *term* pada *cluster*

Term	Df	*idf*
2	1	0,69897
additive	1	0,69897
algoritma	1	0,69897
analisis	5	0
aplikasi	1	0,69897
bahasa	1	0,69897
baiducha	1	0,69897
balancing	1	0,69897
banding	1	0,69897
bangun	1	0,69897
barang	1	0,69897
bas	1	0,69897
bellman-ford	1	0,69897
brawijaya	1	0,69897
budur	1	0,69897
chain	1	0,69897
citra	1	0,69897
connection	1	0,69897
dasar	1	0,69897
data	1	0,69897
defined	1	0,69897
dijkstra	1	0,69897
education	1	0,69897
elite	1	0,69897
enterprise	1	0,69897
evaluasi	1	0,69897
evaluation	1	0,69897
factory	1	0,69897
fakultas	1	0,69897
filkom	1	0,69897
film	1	0,69897
first	1	0,69897
heuristic	1	0,69897
ilmu	1	0,69897
implementasi	1	0,69897
indonesia	1	0,69897
k-nearest	1	0,69897

kamus	1	0,69897
kasus	2	0,39794
kerja	1	0,69897
kernel	1	0,69897
komputer	1	0,69897
lalu	1	0,69897
least	1	0,69897
load	1	0,69897
management	1	0,69897
media	1	0,69897
mesh	1	0,69897
metode	2	0,39794
model	1	0,69897
neighbor	1	0,69897
neighbor-	1	0,69897
network	2	0,39794
networking	1	0,69897
open	1	0,69897
openflow	1	0,69897
opini	1	0,69897
ospf	1	0,69897
pabrik	1	0,69897
path	1	0,69897
pendek	1	0,69897
persona	1	0,69897
planning	1	0,69897
politik	1	0,69897
protokol	1	0,69897
rancang	1	0,69897
resource	1	0,69897
routing	1	0,69897
rute	1	0,69897
sentimen	1	0,69897
server	1	0,69897
shortest	1	0,69897
sistem	1	0,69897
situs	1	0,69897
smkn	1	0,69897
software	1	0,69897
software-defined	1	0,69897
sosial	1	0,69897
studi	1	0,69897
supply	1	0,69897
svm	1	0,69897

technology	1	0,69897
teknologi	1	0,69897
terap	1	0,69897
toys	1	0,69897
traffic	1	0,69897
twitter	1	0,69897
ub	1	0,69897
universitas	1	0,69897
usability	1	0,69897
voice	1	0,69897
web	1	0,69897
weigthed	1	0,69897
wireless	1	0,69897

Tahap selanjutnya yaitu mengalikan tiap-tiap bobot *Tf* dengan nilai *idf.* Tabel 4.21 menunjukan hasil bobot *TF-IDF* dari 2 *cluster.*

Tabel 4.21 Bobot *TF-IDF* pada *cluster*

Term	**C0**	**C1**	**C2**	**C3**	**C4**
2	0	0,69897	0	0	0
additive	0	0	0	0,69897	0
algoritma	0	0	0	0	1.032463
analisis	0	0	0	0	0
aplikasi	0	0,69897	0	0	0
bahasa	0	0	0	0,69897	0
baiducha	0,69897	0	0	0	0
balancing	0	0	0	0	0,909381
banding	0	0	0	0	0,69897
bangun	0	0,69897	0	0	0
barang	0,69897	0	0	0	0
bas	0	0	0	0,69897	0
bellman-ford	0	0	0	0	0,69897
brawijaya	0	0	0	0	0,69897
budur	0	0,69897	0	0	0
chain	0,69897	0	0	0	0
citra	0	0	0	0,69897	0
connection	0	0	0	0	0,909381
dasar	0	0	0	0,69897	0
data	0	0	0,69897	0	0
defined	0	0	0	0	0,909381
dijkstra	0	0	0	0	0,69897
education	0	0,69897	0	0	0
elite	0	0	0	0,69897	0
enterprise	0	0,69897	0	0	0

evaluasi	0	0	0	0	0,69897
evaluation	0	0	0	0	0,69897
factory	0,69897	0	0	0	0
fakultas	0	0	0	0	0,69897
filkom	0	0	0	0	0,69897
film	0	0	0	0,69897	0
first	0	0	0,909381	0	0
heuristic	0	0	0	0	0,69897
ilmu	0	0	0	0	0,69897
implementasi	0	0	0	0	0,909381
indonesia	0	0	0	0,69897	0
k-nearest	0	0	0	0,69897	0
kamus	0	0	0	0,69897	0
kasus	0	0	0,39794	0	0,39794
kerja	0	0	0,69897	0	0
kernel	0	0	0	0,69897	0
komputer	0	0	0	0	0,69897
lalu	0	0	0	0,69897	0
least	0	0	0	0	0,909381
load	0	0	0	0	0,909381
management	0,69897	0	0	0	0
media	0	0	0	0,69897	0
mesh	0	0	0,69897	0	0
metode	0	0	0	0,517732	0,39794
model	0,69897	0	0	0	0
neighbor	0	0	0	0,69897	0
neighbor-	0	0	0	0,69897	0
network	0	0	0,39794	0	0,517732
networking	0	0	0,69897	0	0
open	0	0	0,909381	0	0
openflow	0	0	0	0	0,69897
opini	0	0	0	0,909381	0
ospf	0	0	0,69897	0	0
pabrik	0,69897	0	0	0	0
path	0	0	0,909381	0	0
pendek	0	0	0	0	0,69897
persona	0	0	0	0	0,69897
planning	0	0,69897	0	0	0
politik	0	0	0	0,69897	0
protokol	0	0	0,909381	0	0
rancang	0	0,69897	0	0	0
resource	0	0,69897	0	0	0
routing	0	0	0,909381	0	0
rute	0	0	0	0	0,69897
sentimen	0	0	0	0,909381	0

server	0	0	0	0	0,909381
shortest	0	0	0,909381	0	0
sistem	0,69897	0	0	0	0
situs	0	0	0	0	0,69897
smkn	0	0,69897	0	0	0
software	0	0	0	0	0,909381
software-defined	0	0	0,69897	0	0
sosial	0	0	0	0,69897	0
studi	0	0	0	0	0,69897
supply	0,69897	0	0	0	0
svm	0	0	0	0,69897	0
technology	0,69897	0	0	0	0
teknologi	0	0	0,69897	0	0
terap	0	0	0	0	0,69897
toys	0,69897	0	0	0	0
traffic	0	0	0,69897	0	0
twitter	0	0	0	0,69897	0
ub	0	0	0	0	0,69897
universitas	0	0	0	0	0,69897
usability	0	0	0	0	0,69897
voice	0	0	0,69897	0	0
web	0	0	0	0	1.032463
weigthed	0	0	0	0,69897	0
wireless	0	0	0,69897	0	0

Langkah berikutnya yaitu melakukan pengurutan *term* yang memiliki bobot *TF-IDF* tertinggi dari masing-masing *cluster.* Setelah *term* diurutkan kemudian mengambil 5 *term* teratas sebagai kandidat label dari suatu *cluster.* Tabel 4.22 menunjukan 5 *term* dengan nilai bobot tertinggi dari masing-masing *cluster.*

Tabel 4.22 *Term* dengan bobot tertinggi dari tiap *cluster*

C0		**C1**		**C2**	
***Term* C0**	**Bobot**	***Term* C1**	**Bobot**	***Term* C2**	**Bobot**
chain	0,69897	planning	0,69897	path	0,909381
barang	0,69897	aplikasi	0,69897	protokol	0,909381
factory	0,69897	resource	0,69897	routing	0,909381
sistem	0,69897	rancang	0,69897	first	0,909381
supply	0,69897	bangun	0,69897	open	0,909381
C3		**C4**			
***Term* C3**	**Bobot**	***Term* C4**	**Bobot**		
sentimen	0,909381	web	1.032463		
opini	0,909381	algoritma	1.032463		
neighbor-	0,69897	server	0,909381		
dasar	0,69897	connection	0,909381		
bas	0,69897	load	0,909381		

Tabel diatas menunjukkan kandidat *term* pada tiap *cluster* yang bisa digunakan sebagai label dari *cluster.*

4.3.5 Perhitungan *Silhouette Coefficient (SC).*

Dalam proses perhitungan *silhouette coefficient (SC)* terdapat beberapa tahapan yaitu:

1. Menghitung rata-rata jarak setiap dokumen (disimbolkan dalam *i*) dengan setiap dokumen lain dalam *cluster* yang sama. Simpan nilai rata-rata pada suatu variable misalkan a_i.
2. Menghitung rata-rata nilai jarak setiap dokumen suatu *cluster* (disimbolkan dalam *i*) dengan setiap dokumen pada *cluster* lain dan diambil nilai terkecil. Simpan nilai tersebut pada suatu variable misalkan b_i.
3. Menghitung nilai *Silhouette Coefficient* dari tiap-tiap *cluster.*

Detail rumus perhitungan dari *Silhouette Coefficient (SC)* dapat dilihat pada Sub-Bab 2.8.1 Persamaan 2.7.

Pada manualisasi proses perhitungan *Silhouette Coefficient* kali ini akan mengambil contoh jika *cluster* yang dihasilkan adalah sebanyak 5 *cluster* sesuai dengan pengambilan titik potong tahap 5 seperti pada tahap pelabelan *cluster*.

Tabel 4.23 Nilai *Slihouette Coefficient (SC)*

		a_i	b_i	SC	SC *Cluster*
C0	Doc1	-	-	0	0
C1	Doc2	-	-	0	0
C2	Doc4	0,642379	1	0,357621	0,348424
	Doc7	0,642379	0,972164	0,339228	
C3	Doc8	0,888406	0,994748	0,106904	0,107104
	Doc9	0,888406	0,995196	0,107305	
C4	Doc0	0,713941	1	0,286059	0,179273
	Doc5	0,669902	0,980819	0,316997	
	Doc6	0,893989	0,98281	0,090374	
	Doc3	0,956703	0,979888	0,023661	
			SC Keseluruhan	0,162815	

Nilai *silhouette coefficient* memiliki rentang nilai antara 1 sampai dengan -1. Semakin tinggi atau mendekati 1 berarti dokumen berada *cluster* yang tepat, semakin rendah nilai *silhouette coefficient* (semakin mendekati -1) maka dokumen tidak tepat pada *cluster* tersebut. Nilai *silhouette coefficient* dari *cluster* diperoleh dari nilai rata-rata *silhouette coefficient* dari dokumen pada *cluster* tersebut. Nilai *silhouette coefficient* keseluruhan diperoleh dari rata-rata nilai *silhouette coefficient* dari kesuluruhan dokumen.

4.3.6 Perhitungan *Precision*

Nilai *precision* diperoleh dari label yang didapatkan yang sesuai dengan kata kunci dibagi dengan seluruh label pada suatu *cluster* yang diperoleh. Tabel 4.24 menunjukkan hasil nilai *precision* tiap *cluster.* Detail rumus perhitungan *precision* dapat dilihat pada Sub-Bab 2.9.2 Persamaan 2.7.

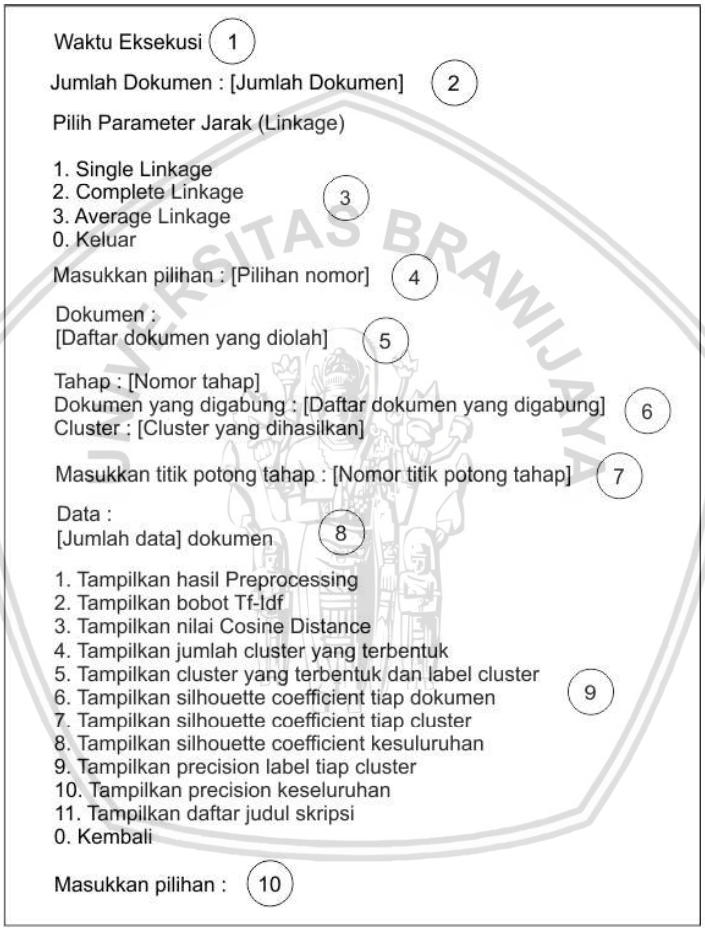
Tabel 4.24 Nilai *Precision* Tiap *Cluster*

Label C0	**Keyword**				**A**	**B**	**Precision**
chain	ada	black-box			3	5	0,6
barang	barang	testing					
factory	supply						
sistem	chain						
supply	management						
Label C1	**Keyword**				**A**	**B**	**Precision**
planning	enterprise	erpe			2	5	0,4
aplikasi	resource	prototyping					
resource	planning	smkn					
rancang	erp	smk					
bangun	education						
Label C2	**Keyword**				**A**	**B**	**Precision**
path	ospf	mesh			0	5	0
protokol	sdn	network					
routing	abilene	wireless					
first	qos	voice					
open	voip						
Label C3	**Keyword**				**A**	**B**	**Precision**
sentimen	analisis	k-nearest	classification	svm	1	5	0,2
opini	sentimen	neighbor	citra	imaging			
neighbor-	klasifikasi	sentiment	tweets				
dasar	teks	analysis	additive				
bas	neighbor-weighted	text	kernel				
Label C4	**Keyword**				**A**	**B**	**Precision**
web	load	web	network	evaluation	5	5	1
algoritma	balancing	least	dijkstra	persona			
server	algoritma	connection	bellman-ford	usability			
connection	jadwal	software	pyretic				
load	server	defined	heuristic				

4.4 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberikan gambaran *output* yang dihasilkan dari *clustering* dokumen skripsi berdasarkan judul dan abstrak dengan menggunakan *Hierarchical Agglomerative Clustering*. Pada penelitian ini, *output* yang dihasilkan oleh program tidak berbentuk *Graphical User Interface* (GUI), melainkan berupa *Command Line Interface* (CLI).

Gambar 4.14 Perancangan antarmuka sistem



Penjelasan perancangan antarmuka system:

1. Menunjukkan waktu eksekusi program.
2. Menunjukkan jumlah dokumen yang diolah.
3. Daftar pilihan jarak sebagai parameter *clustering* dokumen.
4. Perintah memasukkan pilihan parameter jarak.

5. Menampilkan daftar dokumen yang diolah.
6. Menampilkan tahap penggabungan *cluster* dari awal sampai seluruh dokumen menjadi 1 *cluster* beserta dokumen yang sedang digabung dan hasil *cluster* pada tahap tersebut.
7. Perintah untuk memasukkan titik potong tahap untuk mendapatkan hasil *clustering.*
8. Menampilkan jumlah dokumen yang diolah.
9. Menampilkan pilihan menu untuk melihat hasil pengolahan data mulai dari *preprocessing* hingga hasil evaluasi *silhouette coefficient.*
10. Perintah untuk memasukkan pilihan menu.

4.5 Pengujian Metode

Pengujian metode dilakukan untuk melihat nilai hasil evaluasi yaitu *Silhouette Coefficient* terbaik dari parameter-parameter pengujian yang sudah ditentukan. Pengujian yang dilakukan yaitu menggunakan parameter jarak untuk pengelompokan *cluster* yang berbeda yaitu *single linkage, complete linkage* dan *average linkage.* Pada penelitian ini, pengujian metode dengan mengambil beberapa titik potong tahap pembentukan *cluster* yang sudah disimpan. Setiap titik potong akan menghasilkan jumlah *cluster* berbeda-beda.

Tabel 4.25 Rancangan pengujian berdasarkan titik potong pada parameter jarak(*linkage*) yang ditentukan

Parameter Jarak (*Linkage*)		
Titik Potong	Jumlah *Cluster*	Nilai *Silhouette Coefficient*
0	100	
2	98	
4	96	
6	94	
8	92	
10	90	
12	88	
14	86	
16	84	
18	82	
20	80	
22	78	
24	76	
26	74	
28	72	
30	70	
...	...	
98	2	

Setelah seluruh titik potong menghasilkan nilai *silhouette coefficient* maka akan dipilih nilai *silhouette coeffiecient* tertinggi. Nilai *silhouette coefficient* tertinggi dari masing-masing metode pengelompokan (*single linkage, complete linkage, average linkage)* akan dibandingkan untuk menunjukkan parameter jarak mana yang terbaik dalam mengelompokkan dokumen skripsi berdasarkan judul.

Parameter jarak dengan nilai *silhouette coefficient* tertinggi akan dipilih untuk mengetahui label tiap *cluster* yang dihasilkan beserta nilai *silhouette coefficient* tiap dokumen dan *cluster.*

Tabel 4.26 Rancangan hasil pelabelan tiap *cluster*

Titik Potong Dipilih		
Cluster	Dokumen	Label
0		
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		
...		
n		

Tabel di atas menunjukkan hasil pelabelan *cluster* berdasarkan titik potong yang dipilih beserta dokumen yang masuk dalam suatu *cluster.*

Tabel 4.27 Rancangan hasil nilai *Silhouette Coefficient* tiap dokumen

Titik Potong Dipilih		
Cluster	Dokumen	Nilai *Silhouette Coefficient*
0	1	
1	2	
2	3	
3	4	
4	5	
5	6	
6	7	
7	8	

7	9	
7	10	
8	11	
8	12	
8	13	
9	14	
9	15	
9	16	
...	...	
n	n	

Tabel di atas menunjukkan hasil evaluasi dengan *silhouette coefficient* tiap dokumen beserta *cluster* yang memuat dokumen tersebut.

Tabel 4.28 Rancangan hasil nilai Silhouette Coefficient tiap *cluster*

Titik Potong Dipilih		
Cluster	Dokumen	Nilai *Silhouette Coefficient*
0	0	
1	1	
2	2	
3	[3,4]	
4	[5,6]	
5	7	
6	8	
7	9	
8	[10,11,12]	
9	[13,14]	
10	15	
11	16	
12	17	
13	18	
14	19	
15	[20,21]	
...	...	
n	n	

Tabel di atas menunjukkan hasil evaluasi dengan *silhouette coefficient* tiap *cluster* yang diperoleh dari rata-rata nilai *silhouette coefficient* dokumen didalam *cluster* tersebut.

BAB 5 IMPLEMENTASI

Bab ini akan membahas tentang implementasi metode *Hierarchical Agglomerative Clustering (HAC)* untuk *clustering* dokumen skripsi dan antarmuka yang sudah dirancang pada bab 4.

5.1 Implementasi Metode

5.1.1 Implementasi *Preprocessing Data*

```
1   datajudul = pd.read_csv('E:/Duty/Data/dataskripsi.csv')
2   casefold = datajudul.applymap(str.lower)
3   tokenize = casefold.applymap(str.split)
4   tokenizelist = tokenize['Data'].tolist()
5   rangeDoc = 10
6   hasilPreprocessing = []
7   for i in range(0, rangeDoc):
8       isiDoc = []
9       for term in tokenizelist[i]:
10          hasilStem = stemmer.stem(term)
11          hasilStopword = stopword.remove(hasilStem)
12          if hasilStopword:
13              isiDoc.append(hasilStopword)
14      hasilPreprocessing.append(isiDoc)
```

Gambar 5.1 Kode program *preprocessing data*

Baris 1: Membaca file *csv* dokumen skripsi dengan *Pandas dataframe.*

Baris 2: Mengubah kapitalisasi huruf menjadi kecil (*lower*).

Baris 3: Memecah teks menjadi token-token *term* menggunakan fungsi *split*.

Baris 4: Memasukkan hasil tokenisasi dari seluruh judul ke *list*.

Baris 5: Mendeklarasikan jumlah dokumen yang akan dilakukan *clustering.*

Baris 10: Melakukan proses *stemming.*

Baris 11: Menghapus *term* yang berupa *stopword.*

Baris 14: Menyimpan hasil akir dari *preprocessing data.*

5.1.2 Implementasi *Pembobotan TF-IDF*

5.1.2.1 Implementasi Perhitungan *Tf*

```
1   fDoc = []
2   checkTerm = {}
3   for x in range(0,rangeDoc):
4       fisiDoc = {}
5       freq_temp = {}
6       for z in hasilPreprocessing[x]:
7           count = freq_temp.get(z,0)
8           freq_temp[z] = count + 1
9           fisiDoc[z] = freq_temp[z]
10          checkTerm[z] = 1
11      fDoc.append(fisiDoc)
12  listAllTerm = checkTerm.keys()
13  tfDocAll = []
14  tfd = {}
15  freqDocBaru = {}
16  fDocBaru = []
17  for x in range (0, rangeDoc):
18      tfisid = {}
19      fisiDocBaru = {}
20      for word in listAllTerm:
21          if word in hasilPreprocessing[x]:
22              f_temp = fDoc[x]
23              tfd[word] = 1 + math.log10(f_temp[word])
24              tfisid[word] = tfd[word]
25              freqDocBaru[word] = f_temp[word]
26              fisiDocBaru[word] = freqDocBaru[word]
27          else:
28              tfd[word] = 0
29              tfisid[word] = tfd[word]
30              freqDocBaru[word] = 0
31              fisiDocBaru[word] = freqDocBaru[word]
32      tfDocAll.append(tfisid)
33      fDocBaru.append(fisiDocBaru)
```

Gambar 5.2 Kode program perhitungan *Tf*

Baris 3-11 : Menghitung frekuensi *term* pada tiap-tiap dokumen dan melakukan penandaan bahwa *term* tersebut telah muncul.

Baris 12 : Menyimpan seluruh *term* yang muncul pada seluruh dokumen.

Baris 21-26 : Jika *term* terdapat pada dokumen maka akan dihitung nilai bobot Tf.

Baris 27-31 : Jika *term* tidak ada pada dokumen maka nilai bobot Tf = 0,

5.1.2.2 Implementasi Perhitungan *idf*

```
1   df = {}
2   dfterm = {}
3   idf = {}
4   for word in listAllTerm:
5       for index in range(0, rangeDoc):
6           if word in hasilPreprocessing[index]:
7               cn = df.get(word,0)
8               df[word] = cn + 1
9               dfterm[word] = df[word]
10              ndf = float(rangeDoc)/float(df[word])
11              idf[word] = math.log10(ndf)
```

Gambar 5.3 Kode program perhitungan *idf*

Baris 6-11 : Menghitung nilai *idf* jika *term* yang dipilih muncul pada dokumen yang dipilih.

5.1.2.3 Implementasi Perhitungan *TF-IDF*

```
1   tfidfDoc = []
2   for i in range(0,rangeDoc):
3       valTfidf = {}
4       tflist_temp = tfDocAll[i]
5       for termDoc in tflist_temp.keys():
6           if termDoc in idf.keys():
7               tfidf = idf.get(termDoc) *
    tflist_temp.get(termDoc)
8               valTfIdf[termDoc] = tfidf
9       tfidfDoc.append(valTfIdf)
```

Gambar 5.4 Kode program perhitungan *TF-IDF*

Baris 7-8: Mengalikan bobot Tf dengan nilai *idf* setiap *term* pada tiap-tiap dokumen yang dipilih.

5.1.3 Implementasi *Cosine Distance*

```
1   temp_powTfIdf = []
2   for i in range (0, rangeDoc):
3       temp_powWtd = {}
4       powWtd = {}
5       temp_tfidfDocValue = tfidfDoc[i]
6       for j in temp_tfidfDocValue:
7           powTfIdf = math.pow(temp_tfidfDocValue[j], 2)
8           temp_powWtd[j] = powTfIdf
9       temp_powTfIdf.append(temp_powWtd)
10
11  temp_DotProdDoc = []
12  for i in range (0,rangeDoc):
13      temp_DotProd = []
14      for j in range (0,rangeDoc):
15          temp_DotProdTerm = {}
16          temp_TfIdf_1 = tfidfDoc[i]
17          temp_TfIdf_2 = tfidfDoc[j]
18          temp_powTfIdf_1 = temp_powTfIdf[i]
19          temp_powTfIdf_2 = temp_powTfIdf[j]
20          for term1 in temp_powTfIdf_1:
```

```
21              for term2 in temp_powTfIdf_2:
22                  if (term1 == term2):
23                      dotprod = temp_TfIdf_1[term1] *
    temp_Tfidf_2[term1]
24                      temp_DotProdTerm[term1] = dotprod
25          temp_DotProd.append(temp_DotProdTerm)
26      temp_DotProdDoc.append(temp_DotProd)
27
28  CosDisDoc = []
29  for i in range (0,rangeDoc):
30      temp_CosSimDoc = []
31      temp_CosDisDoc = []
32      for j in range (0,rangeDoc):
33          DotProdDoc = temp_DotProdDoc[i][j]
34          temp_powTfIdf_1 = temp_powTfIdf[i]
35          temp_powTfIdf_2 = temp_powTfIdf[j]
36          CosSim = round(sum(DotProdDoc.values()) /
    math.sqrt(sum(temp_powTfIdf_1.values())*sum(temp_powTfIdf_2
    .values())),15)
37          CosDis = 1-CosSim
38          temp_CosDisDoc.append(CosDis)
39      CosDisDoc.append(temp_CosDisDoc)
```

Gambar 5.5 Kode program *cosine distance*

Baris 6-8 : Menghitung nilai kuadrat dari bobot *TF-IDF* masing-masing *term* pada setiap dokumen.

Baris 20-23 : Menghitung hasil kali bobot *TF-IDF* masing-masing *term* dari 2 dokumen yang dipilih.

Baris 36 : Perhitungan nilai *cosine similarity.*

Baris 37 : Perhitungan nilai *cosine distance.*

5.1.4 Implementasi Pemilihan Dokumen yang Digabung Dengan *Single Linkage*

```
1   P = len(CosDisDoc)
2   cluster = []
3   for i in range (0, P):
4       cluster.append(i)
5   gabung = []
6   thp = 0
7   tahap = {}
8   while (len(cluster)!=1):
9       if thp == 0:
10          print "Tahap : ",thp
11          print "Cluster : ",cluster
12          tahap[thp]=copy.deepcopy(cluster)
13      else:
14          print "Tahap : ",thp
15          listCosDis = {}
16          #SINGLE LINKAGE
17          if pilihlinkage == 1:
18              for doc1 in  cluster:
19                  for doc2 in cluster:
20                      if doc1 != doc2:
```

```
21                          if isinstance(doc1, (list))==False
    and isinstance(doc2, (list))==False :
22                              nilaiCosDis =
    CosDisDoc[doc1][doc2]
23                              listCosDis[doc1,doc2] =
    nilaiCosDis
24                          elif isinstance(doc1,
    (list))==False and isinstance(doc2, (list))==True :
25                              newGrup = {}
26                              for doc in doc2:
27                                  nilaiSementara =
    CosDisDoc[doc1][doc]
28                                  newGrup[doc1,doc] =
    nilaiSementara
29                              urutanGrup = [(value, key) for
    key, value in newGrup.items()]
30                              ambilDokumen =
    min(urutanGrup)[1]
31                              for doc1_grup in ambilDokumen:
32                                  for doc2_grup in
    ambilDokumen:
33                                      if doc1_grup !=
    doc2_grup:
34                                          nilaiCosDis =
    CosDisDoc[doc1_grup][doc2_grup]

35  listCosDis[doc1_grup,doc2_grup] = nilaiCosDis
36                          elif isinstance(doc1, (list))==True
    and isinstance(doc2, (list))==True :
37                              newGrup = {}
38                              for doc_1 in doc1:
39                                  for doc_2 in doc2:
40                                      nilaiSementara =
    CosDisDoc[doc_1][doc_2]
41                                      newGrup[doc_1,doc_2] =
    nilaiSementara
42                              urutanGrup = [(value, key) for
    key, value in newGrup.items()]
43                              ambilDokumen =
    min(urutanGrup)[1]
44                              for doc1_grup in ambilDokumen:
45                                  for doc2_grup in
    ambilDokumen:
46                                      if doc1_grup !=
    doc2_grup:
47                                          nilaiCosDis =
    CosDisDoc[doc1_grup][doc2_grup]

48  listCosDis[doc1_grup,doc2_grup] = nilaiCosDis
49          urutanCosDis = [(value, key) for key, value in
    listCosDis.items()]
50          dokDiambil = min(urutanCosDis)[1]
```

Gambar 5.6 Kode program pemilihan dokumen yang digabung dengan *Single linkage*

Baris 3-4 : Inisialisasi daftar dokumen yang diolah ke dalam bentuk *list*.

Baris 8 : Kondisi apabila seluruh data pada *cluster* belum tergabung menjadi 1 maka akan menjalankan proses pengelompokan data.

Baris 9-12 : Menampilkan *cluster* tahap awal yaitu dokumen-dokumen masih bersifat individua tau *singleton.*

Baris 21-23 : Mengambil nilai *cosine distance* dari 2 dokumen apabila element yang diambil keduanya bukan suatu *list* atau *cluster* yang artinya 2 elemen tersebut merupakan 2 dokumen.

Baris 24-35 : Mengambil nilai *cosine distance* apabila ditemukan antara 2 element dalam *cluster* salah satunya berupa *list* atau *cluster* maka elemen yang *list* atau *cluster* tersebut kembali dibuka setiap elemennya sehingga dokumen 1 dengan dokumen lain didalam *list* atau *cluster* diambil nilai *cosine distance* minimumnya untuk disimpan.

Baris 36-48 : Mengambil nilai *cosine distance* apabila ditemukan antara 2 element dalam *cluster* keduanya berupa *list* atau *cluster* maka setiap elemen yang *list* atau *cluster* tersebut kembali dibuka setiap elemennya sehingga tiap dokumen pada *cluster* satu dengan tiap dokumen pada *cluster* satunya diambil nilai *cosine distance* minimumnya untuk disimpan.

Baris 49-50 : Mengambil 2 dokumen dengan nilai terkecil dari daftar nilai *cosine distance* untuk kemudian digabungkan.

5.1.5 Implementasi Pemilihan Dokumen yang Digabung Dengan *Complete Linkage*

```
1   #COMPLETE LINKAGE
2   elif pilihlinkage == 2:
3       for doc1 in  cluster:
4           for doc2 in cluster:
5               if doc1 != doc2:
6                   if isinstance(doc1, (list))==False and
    isinstance(doc2, (list))==False :
7                       nilaiCosDis = CosDisDoc[doc1][doc2]
8                       listCosDis[doc1,doc2] = nilaiCosDis
9                       elif isinstance(doc1, (list))==False
    and isinstance(doc2, (list))==True :
10                          newGrup = {}
11                          for doc in doc2:
12                              nilaiSementara =
    CosDisDoc[doc1][doc]
13                              newGrup[doc1,doc] =
    nilaiSementara
14                          urutanGrup = [(value, key) for key,
    value in newGrup.items()]
15                          ambilDokumen = max(urutanGrup)[1]
16                          for doc1_grup in ambilDokumen:
17                              for doc2_grup in ambilDokumen:
18                                  if doc1_grup != doc2_grup:
19                                      nilaiCosDis =
    CosDisDoc[doc1_grup][doc2_grup]
20
```

```
21      listCosDis[doc1_grup,doc2_grup] = nilaiCosDis
22                          elif isinstance(doc1, (list))==True and
        isinstance(doc2, (list))==True :
23                              newGrup = {}
24                              for doc_1 in doc1:
25                                  for doc_2 in doc2:
26                                      nilaiSementara =
        CosDisDoc[doc_1][doc_2]
27                                      newGrup[doc_1,doc_2] =
        nilaiSementara
28                              urutanGrup = [(value, key) for key,
        value in newGrup.items()]
29                              ambilDokumen = max(urutanGrup)[1]
30                              for doc1_grup in ambilDokumen:
31                                  for doc2_grup in ambilDokumen:
32                                      if doc1_grup != doc2_grup:
                                            nilaiCosDis =
33      CosDisDoc[doc1_grup][doc2_grup]

34      listCosDis[doc1_grup,doc2_grup] = nilaiCosDis
35              urutanCosDis = [(value, key) for key, value in
        listCosDis.items()]
36              dokDiambil = min(urutanCosDis)[1]
```

Gambar 5.7 Kode program pemilihan dokumen yang digabung dengan *complete linkage*

Baris 6-8 : Mengambil nilai *cosine distance* dari 2 dokumen apabila element yang diambil keduanya bukan suatu *list* atau *cluster* yang artinya 2 elemen tersebut merupakan 2 dokumen.

Baris 9-21 : Mengambil nilai *cosine distance* apabila ditemukan antara 2 element dalam *cluster* salah satunya berupa *list* atau *cluster* maka elemen yang *list* atau *cluster* tersebut kembali dibuka setiap elemennya sehingga dokumen 1 dengan dokumen lain didalam *list* atau *cluster* diambil nilai *cosine distance* maksimumnya untuk disimpan.

Baris 22-34 : Mengambil nilai *cosine distance* apabila ditemukan antara 2 element dalam *cluster* keduanya berupa *list* atau *cluster* maka setiap elemen yang *list* atau *cluster* tersebut kembali dibuka setiap elemennya sehingga tiap dokumen pada *cluster* satu dengan tiap dokumen pada *cluster* satunya diambil nilai *cosine distance* maksimumnya untuk disimpan.

Baris 35-36 : Mengambil 2 dokumen dengan nilai terkecil dari daftar nilai *cosine distance* untuk kemudian digabungkan.

5.1.6 Implementasi Pemilihan Dokumen yang Digabung Dengan *Average Linkage*

```
1   #AVERAGE LINKAGE
2   elif pilihlinkage == 3:
3       for doc1 in  cluster:
4           for doc2 in cluster:
5               if doc1 != doc2:
6                   if isinstance(doc1, (list))==False and
    isinstance(doc2, (list))==False :
7                       nilaiCosDis = CosDisDoc[doc1][doc2]
8                       listCosDis[doc1,doc2] = nilaiCosDis
9                       elif isinstance(doc1, (list))==False
    and isinstance(doc2, (list))==True :
10                          averageCosDis = []
11                          for doc in doc2:
12
    averageCosDis.append(CosDisDoc[doc1][doc])
13                          averageCosDisValue =
    np.mean(averageCosDis)
14                          for doc in doc2:
15                              listCosDis[doc1,doc] =
    averageCosDisValue
16                      elif isinstance(doc1, (list))==True and
    isinstance(doc2, (list))==True :
17                          averageCosDis = []
18                          for doc_1 in doc1:
19                              for doc_2 in doc2:
20  averageCosDis.append(CosDisDoc[doc_1][doc_2])
21                          averageCosDisValue =
    np.mean(averageCosDis)
22                          for doc_1 in doc1:
23                              for doc_2 in doc2:
24                                  listCosDis[doc_1,doc_2] =
    averageCosDisValue
25          urutanCosDis = [(value, key) for key, value in
    listCosDis.items()]
26          dokDiambil = min(urutanCosDis)[1]
```

Gambar 5.8 Kode program pemilihan dokumen yang digabung dengan *average linkage*

Baris 6-8 : Mengambil nilai *cosine distance* dari 2 dokumen apabila element yang diambil keduanya bukan suatu *list* atau *cluster* yang artinya 2 elemen tersebut merupakan 2 dokumen.

Baris 9-15 : Mengambil nilai *cosine distance* apabila ditemukan antara 2 element dalam *cluster* salah satunya berupa *list* atau *cluster* maka elemen yang *list* atau *cluster* tersebut kembali dibuka setiap elemennya sehingga dokumen 1 dengan dokumen lain didalam *list* atau *cluster* diambil nilai rata-rata *cosine distance* dari 2 dokumen untuk disimpan.

Baris 16-24 : Mengambil nilai *cosine distance* apabila ditemukan antara 2 element dalam *cluster* keduanya berupa *list* atau *cluster* maka setiap elemen yang *list* atau *cluster* tersebut kembali dibuka setiap elemennya sehingga tiap dokumen

pada *cluster* satu dengan tiap dokumen pada *cluster* satunya diambil nilai rata-rata *cosine distance* antara 2 dokumen untuk disimpan.

Baris 25-26 : Mengambil 2 dokumen dengan nilai terkecil dari daftar nilai *cosine distance* untuk kemudian digabungkan.

5.1.7 Implementasi Penggabungan dan *Cluster*

```
1   gabunganDokumen = []
2   dictemp = {}
3   print "Doc digabung :",dokDiambil
4   combineSingleton = []
5   cek = []
6   for dokumen in dokDiambil:
7       gabunganDokumen.append(dokumen)
8       dictemp[dokumen] = int(dokumen)
9       for objek in cluster:
10          if isinstance(objek, (list)) :
11              for doclist in objek:
12                  for doctemp in gabunganDokumen:
13                      if doclist==doctemp:
14                          cek.append(doclist)
15          elif isinstance(objek, (list)) == False:
16             if dokumen == objek:
17                combineSingleton.append(dokumen)
18      if len(combineSingleton) == 2:
19          for x in dokDiambil:
20              cluster.remove(x)
21          cluster.append(gabunganDokumen)
22      for dokumen in dokDiambil:
23          if len(cek)>0:
24              gabungClust = []
25              hapusClust = []
26              temp_cek = []
27              doc = 0
28              cluster_cek = []
29              cekclust = 0
30              for objek in cluster:
31                  objekberupaList = isinstance(objek,
    (list,))
32                  if objekberupaList :
33                      for dokumen_padaDokDiambil in
    dokDiambil:
34                          if dokumen_padaDokDiambil not in
    objek:
35                              if dokumen_padaDokDiambil in
    cluster:
36                                  doc =
    dokumen_padaDokDiambil
37                                  for grup in cluster:
38                                      if isinstance(grup,
    (list,)) :
39                                          for d in
    dokDiambil:
40                                              if d in grup
    and d not in cluster:
41
42  grup.append(dokumen_padaDokDiambil)
```

```
43                                                          for doc1 in
44  gabungClust:
                                                                if doc1
45  in grup:

46  cekclust+=1

47  cluster.remove(dokumen_padaDokDiambil)
                                  elif dokumen_padaDokDiambil not
48  in cluster:
                                      for elementCluster in
49  cluster:
                                          if
50  isinstance(elementCluster, (list,)):
                                              if
51  dokumen_padaDokDiambil in elementCluster:
                                                  if
52  dokumen_padaDokDiambil not in temp_cek:

53  temp_cek.append(dokumen_padaDokDiambil)
                                                  if
54  len(temp_cek) <= len(gabunganDokumen):
                                                      if
55  cluster.index(elementCluster) not in hapusClust:

56  hapusClust.append(cluster.index(elementCluster))
                                                      for
57  semuaDokumen in elementCluster:
                                                          if
58  semuaDokumen not in gabungClust:

59  gabungClust.append(semuaDokumen)
60
61              if len(gabungClust)!=0:
62                  cluster.append(gabungClust)
63              urutanHapus = [x for n, x in enumerate(cluster)
64  if x in cluster[:n]]
                if len(urutanHapus)!=0:
65                  for x in urutanHapus:
66                      cluster.remove(x)
67              else:
68                  for x in sorted(hapusClust, reverse=True):
69                      del cluster[x]
70      print "Cluster : ",cluster
71      tahap[thp]=copy.deepcopy(cluster)
72  thp+=1
73
```

Gambar 5.9 Kode program penggabungan dan *cluster*

Baris 6 : Mengambil seluruh dokumen yang *term*asuk dalam dokumen yang digabung satu per satu.

Baris 10-14 : Mengambil setiap dokumen pada tiap *cluster* apabila terdapat dokumen pada *cluster* yang sama dengan dokumen yang digabung maka dokumen akan dimasukkan ke variable cek. Variabel cek berfungsi untuk mengecek jika 2 *cluster* memang harus digabung.

Baris 15-17 : Menggabungkan 2 dokumen yang bersifat *singleton*.

Baris 18-21 : Menghapus 2 dokumen yang akan digabung dan menambahkan *cluster* dari 2 dokumen tersebut ke daftar *cluster*.

Baris 30-47 : Melakukan penggabungan apabila 2 dokumen yang digabung salah satunya merupakan singleton dan salah satunya lagi terdapat pada suatu *cluster* maka menggabungkan dokumen tersebut dengan *cluster* tersebut yang dipilih.

Baris 48-63 : Melakukan proses penggabungan apabila 2 dokumen yang digabung terdapat pada 2 *cluster* yang berbeda maka kedua *cluster* tersebut akan digabungkan.

Baris 62-67 : Menghapus dokumen yang telah digabung ke dalam *cluster*.

Baris 68-70 : Menghapus *cluster* yang sudah digabungkan dengan *cluster* lain.

Baris 72 : Menyimpan hasil *update cluster* baru yang terbentuk ke variable tahap. Variabel tahap digunakan untuk titik potong dalam mengambil hasil *cluster* dari *Hierarchical Agglomerative Clustering.*

5.1.8 Implementasi Mengambil Titik Potong

```
1  inputStep = input("Masukkan titik potong langkah : ")
2  step = inputStep
```

Gambar 5.10 Kode program mengambil titik potong

Variabel inputStep akan menyimpan nilai titik potong yang dimasukkan oleh pengguna. Variabel step menyimpan nilai dari inputStep. Variabel step merupakan titik potong yang digunakan untuk mengambil *cluster* yang dihasilkan berdasarkan tahap-tahap yang dilalui dalam proses pembentukan *cluster*.

5.1.9 Implementasi Menampilkan Label *Cluster*

```
1  tfCluster = []
2  for i in range(0, len(tahap[step])):
3      if isinstance(tahap[step][i], (list,)):
4          tfClusterList = {}
5          for doc in tahap[step][i]:
6              for x in fDocBaru[doc]:
7                  if x in tfClusterList:
8                      tfClusterList[x] = tfClusterList[x] +
   fDocBaru[doc][x]
9                  else:
10                     tfClusterList[x] = fDocBaru[doc][x]
           tfCluster.append(tfClusterList)
11     elif isinstance(tahap[step][i], (list,))==False:
12         tfCluster.append(fDocBaru[tahap[step][i]])
13 #Bobot TF Cluster
14 WtfCluster = []
15 for i in range (0, len(tfCluster)):
16     WtfTerm = {}
17     for x in tfCluster[i]:
18         freq = int(tfCluster[i][x])
19         if freq != 0 :
20             WtfTerm[x] = 1 + math.log10(freq)
```

```
21          else:
22              Wtf_Term[x] = 0
23      Wtf_Cluster.append(Wtf_Term)
24
25  #MENGHITUNG IDF CLUSTER
26  tempClus = {}
27  dfClus = {}
28  df_termClus = {}
29  idfClus = {}
30  for word in _listAll_Term:
31      for index in range(0, len(tf_Cluster)):
32          if word in tf_Cluster[index]:
33              if tf_Cluster[index][word] != 0:
34                  cn = dfClus.get(word,0)
35                  dfClus[word] = cn + 1
36                  df_termClus[word] = dfClus[word]
37                  ndfClus =
    float(len(tf_Cluster))/float(dfClus[word])
38                  idfClus[word] = math.log10(ndfClus)
39
40  #PEMBOBOTAN TF-IDF CLUSTER
41  tf_idf_clust = []
42  for i in range(0,len(tf_Cluster)):
43      valTfIdfClus = {}
44      tf_list_temp = Wtf_Cluster[i]
45      for _termDoc in tf_list_temp.keys():
46          if _termDoc in idfClus.keys():
47              tfidfClus = idfClus.get(_termDoc) *
    tf_list_temp.get(_termDoc)
48              valTfIdfClus[_termDoc] = tfidfClus
    tf_idf_clust.append(valTfIdfClus)
49
50  label = {}
51  for i in range(0,len(tf_idf_clust)):
52      valLabel = []
53      sorting = sorted(tf_idf_clust[i],
    key=tf_idf_clust[i].get, reverse=True)
54      no = 0
55      for r in sorting:
56          if no > 4 :
57              break
58          else:
59              no = no + 1
60              valLabel.append(r)
61      label[str(tahap[step][i])] = valLabel
```

Gambar 5.11 Kode program pelabelan *cluster*

Baris 2-12 : Menghitung frekuensi *term* yang dihasilkan dari tiap-tiap *cluster*. Frekuensi *term* tiap *cluster* didapat dari menjumlahkan seluruh frekuensi *term* dari dokumen-dokumen yang terdapat pada *cluster*. Apabila *cluster* bersifat singleton maka akan tetap menggunakan frekuensi *term* dari dokumen tersebut.

Baris 15-23 : Menghitung bobot Tf dari tiap-tiap *cluster*.

Baris 25-38 : Menghitung nilai *idf* dari *term* yang muncul dari seluruh *cluster*.

Baris 42-48 : Menghitung bobot *TF-IDF* seluruh *term* dari masing-masing *cluster*.

Baris 50-60 : Mengurutkan bobot *TF-IDF* dari tertinggi ke terendah dari masing-masing *cluster* dan mengambil 5 bobot *TF-IDF* tertinggi dari masing-masing *cluster*.

Baris 61 : Menyimpan label *cluster* untuk masing-masing *cluster*.

5.1.10 Implementasi Evaluasi *Silhouette Coefficient*

```
#Evaluasi
sum_temp_bi = 0
bi_sort = []
testes = []
bi_sort_doc = []
sc_cls = {}
ai_doc = {}
bi_doc = {}
sc_doc = {}
for i in range(0, len(tahap[step])):
    if isinstance(tahap[step][i], (list,)):
        #ai
        for doc in tahap[step][i]:
            ai_bag = []
            for doc_pembanding in tahap[step][i]:
                if doc != doc_pembanding:

ai_bag.append(CosDisDoc[doc][doc_pembanding])
            ai = np.mean(ai_bag)
            ai_doc[doc] = ai

for doc1 in tahap[step]:
    if isinstance(doc1, (list)):
        for docclust1 in doc1:
            bisorttes = []
            for doc2 in tahap[step]:
                if isinstance(doc2, (list)):
                    if doc1 != doc2:
                        bi_bag_list = []
                        for docclust2 in doc2:

bi_bag_list.append(CosDisDoc[docclust1][docclust2])
                        bi_mean = np.mean(bi_bag_list)
                        bisorttes.append(bi_mean)
                elif isinstance(doc2, (list))==False:

bisorttes.append(CosDisDoc[docclust1][doc2])
            if len(bisorttes)!=0:
                bi = min(bisorttes)
                bi_doc[docclust1] = bi

for doc_ai in ai_doc:
    for doc_bi in bi_doc:
        if doc_ai == doc_bi:
            sc_doc[doc_ai] = (bi_doc[doc_bi] -
ai_doc[doc_ai]) / max(bi_doc[doc_bi], ai_doc[doc_ai])
            sc_cls[str(doc_ai)] = sc_doc[doc_ai]
            for i in range(0, len(tahap[step])):
                if isinstance(tahap[step][i],
(list,))==False:
                    sc_cls[tahap[step][i]] = 0
nomorCluster = {}
```

```
49  for i in range(0,len(tahap[step])):
50      nomorCluster[i] = tahap[step][i]
51  #SC tiap cluster
52  sc_avg_cls = {}
53  for i in nomorCluster:
54      if isinstance(nomorCluster[i], (list,)):
55          avgCls = []
56          for doc in nomorCluster[i]:
57              if len(sc_cls)!=0:
58                  avgCls.append(sc_cls[str(doc)])
59              else:
60                  avgCls.append(0)
61          avgSC = np.mean(avgCls)
62          sc_avg_cls[i] = avgSC
63      else:
64          if len(sc_cls)!=0:
65              sc_avg_cls[i] = sc_cls[nomorCluster[i]]
66          else:
67              sc_avg_cls[i] = 0
68  #SC keseluruhan
69  if len(sc_cls)!=0:
70      sc_all = np.mean(sc_cls.values())
71  else:
72      sc_all = 0
```

Gambar 5.12 Kode program evaluasi dengan *silhouette coefficient*

Baris 10-19 : Perhitungan rata-rata nilai *cosine distance* dari suatu dokumen dengan seluruh dokumen dalam 1 *cluster* yang sama (a_i).

Baris 21-37 : Perhitungan rata-rata nilai *cosine distance* dari suatu dokumen dengan setiap dokumen dari *cluster* lain dan diambil nilai terkecil dari rata-rata *cosine distance* dari setiap *cluster* (b_i).

Baris 39-48 : Menghitung *silhouette coeffiecient* dari tiap-tiap dokumen, apabila terdapat *singleton* maka nilai *silhouette coefficient* adalah 0,

Baris 53-67 : Menghitung *silhouette coefficient* tiap-tiap *cluster* yang didapat dari mengambil rata-rata nilai *silhouette coefficient* seluruh dokumen yang terdapat pada masing-masing *cluster,* apabila *cluster* adalah *singleton* maka nilai *silhouette coefficient cluster* tersebut adalah 0,

Baris 69-72 : Menghitung *silhouette coefficient* keseluruhan yang dihasilkan dari seluruh dokumen pada titik potong tahap yang dipilih dengan menghitung rata-rata seluruh nilai *silhouette coefficient.*

5.1.11 Implementasi Evaluasi *Precision*

```
1   keywordAll = {}
2       for i in range(0, len(tahap[step])):
3           if isinstance(tahap[step][i], (list,)):
4               keywordCluster = []
5               for doc in tahap[step][i]:
6                   for term in list_keyword[doc]:
7                       if term not in keywordCluster:
8                           keywordCluster.append(term)
9               keywordAll[str(tahap[step][i])] =
    keywordCluster
10          elif isinstance(tahap[step][i], (list,))==False:
11              keywordCluster = []
12              for term in list_keyword[tahap[step][i]]:
13                  if term not in keywordCluster:
14                      keywordCluster.append(term)
15              keywordAll[str(tahap[step][i])] =
    keywordCluster
16      labelisKeyword = {}
17      for cluster in label:
18          evaluasiLabelCluster = []
19          for term in label[cluster]:
20              if term in keywordAll[cluster]:
21                  evaluasiLabelCluster.append(term)
22          labelisKeyword[cluster] = evaluasiLabelCluster
23
24      hasilPrecisionLabel = {}
25      for cluster in labelisKeyword:
26          a = float(len(labelisKeyword[cluster]))
27          b = float(len(label[cluster]))
28          precisionLabelTiapCluster = (a / b)
29          hasilPrecisionLabel[cluster] =
    precisionLabelTiapCluster
30
31      rataPrecision = np.mean(hasilPrecisionLabel.values())
```

Gambar 5.13 Kode program evaluasi dengan *precision*

Baris 1-15 : Memasukkan label dari tiap-tiap kata kunci dari dokumen ke dalam suatu *cluster.*

Baris 16-22 : Proses untuk mendapatkan label yang diperoleh yang relevan dengan kata kunci.

Baris 24-29 : Proses perhitungan *precision.*

Barus 31 : Menghitung rata-rata nilai *precision* yang diperoleh dari seluruh *cluster.*

5.2 Implementasi Antarmuka

Implementasi dari perancangan antarmuka yang berbentuk *Command Line Interface* (CLI) pada bab perancangan dapat dilihat pada gambar 5.14.

Gambar 5.14 Implementasi Antarmuka

BAB 6 PENGUJIAN DAN ANALISIS

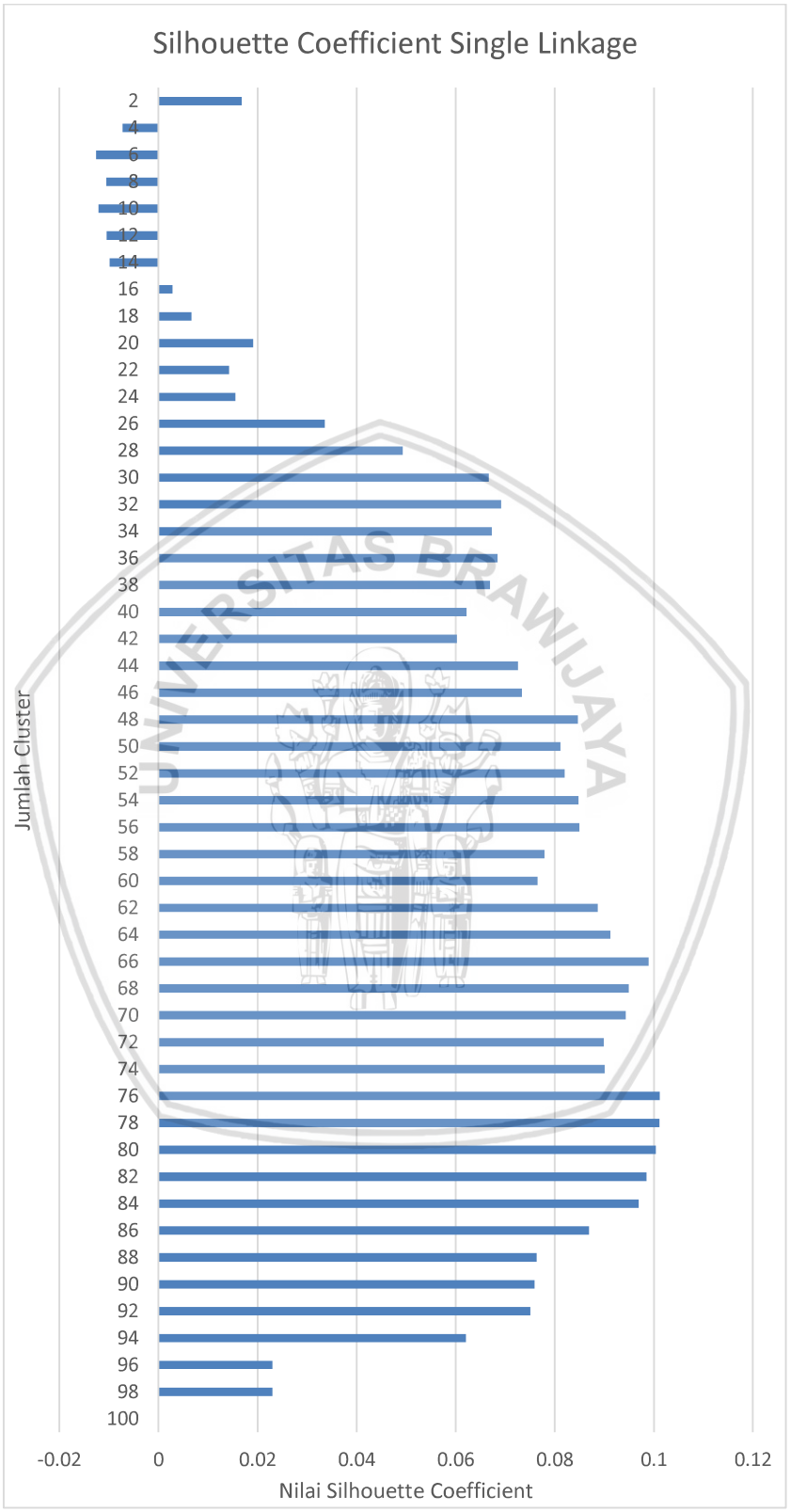
Bab ini akan membahas tentang pengujian dan analisis hasil *clustering* dokumen skripsi berdasarkan abstrak dan judul dengan menggunakan *Hierarchical Agglomerative Clustering (HAC).* Tujuan dari pengujian dan analisis ini adalah mengetahui tingkat ketepatan dokumen dalam suatu *cluster*.

6.1 Pengujian Berdasarkan Titik Potong Pada *Single Linkage*

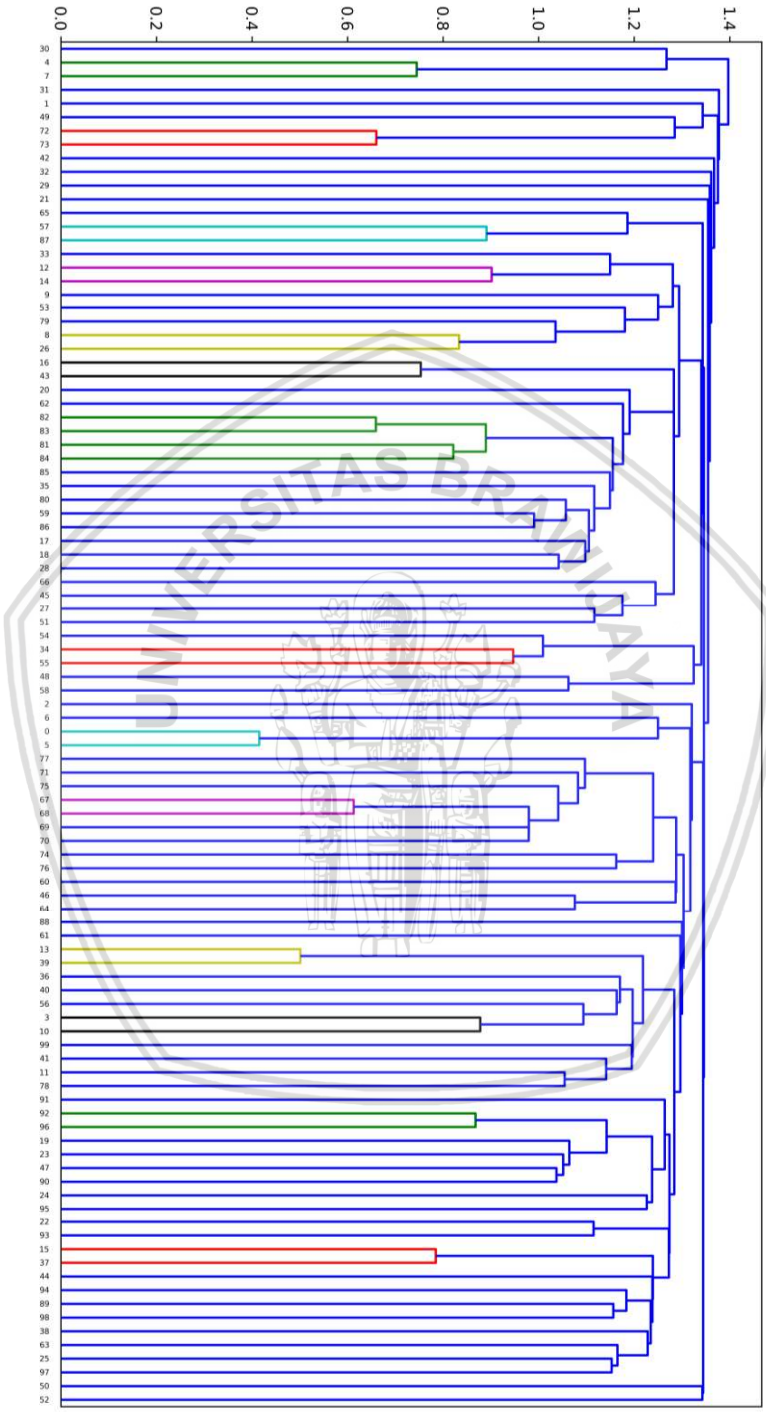
Tabel 6.1 menunjukkan hasil pengujian nilai *Silhouette Coefficient (SC)* dari parameter jarak *complete linkage* yang dihasilkan berdasarkan titik potong yang menghasilkan jumlah *cluster* mulai dari 100 *cluster* sampai dengan 2 *cluster*. Setiap titik potong yang diambil bertambah 2 sehingga jumlah *cluster* yang dihasilkan berkurang 2 setiap titik potongnya.

Tabel 6.1 Pengujian berdasarkan titik potong tiap tahap dengan *single linkage*

Single Linkage			*Single Linkage*		
Titik Potong	Jumlah *Cluster*	SC	Titik Potong	Jumlah *Cluster*	SC
0	100	0	50	50	0,081165
2	98	0,022971	52	48	0,084732
4	96	0,022971	54	46	0,073403
6	94	0,062094	56	44	0,07257
8	92	0,075112	58	42	0,060222
10	90	0,075933	60	40	0,062218
12	88	0,076371	62	38	0,06692
14	86	0,086923	64	36	0,068504
16	84	0,096957	66	34	0,067272
18	82	0,098532	68	32	0,06919
20	80	0,100425	70	30	0,066704
22	78	0,101156	72	28	0,049352
24	76	0,10125	74	26	0,033546
26	74	0,090084	76	24	0,015529
28	72	0,089903	78	22	0,014251
30	70	0,094357	80	20	0,01907
32	68	0,094958	82	18	0,006709
34	66	0,098932	84	16	0,002871
36	64	0,091233	86	14	-0,00988
38	62	0,088693	88	12	-0,01043
40	60	0,076537	90	10	-0,01212
42	58	0,077915	92	8	-0,01049
44	56	0,08499	94	6	-0,01254
46	54	0,084831	96	4	-0,00724
48	52	0,082013	98	2	0,016805



Gambar 6.1 Grafik nilai *Silhouette Coefficient Single Linkage*



Gambar 6.2 Dendrogram *Single Linkage*

Gambar 6.2 menunjukkan hasil dendrogram pengelompokan dokumen dengan parameter *single linkage.*

Bardasarkan Tabel 6.1 dan Gambar 6.1 mengenai hasil pengelompokan dokumen menggunakan parameter jarak *single linkage* diperoleh nilai *silhouette coefficient* tertinggi pada titik potong 24 dengan jumlah *cluster* sebanyak 76 *cluster.* Nilai rata-rata *silhouette coefficient* dari seluruh dokumen yang dihasilkan pada titik potong 24 yaitu 0,10125. Titik potong 24 memiliki nilai *silhouette coefficient* tertinggi yang artinya seluruh dokumen berada pada *cluster* yang tepat atau dikelompokkan dengan tepat dibandingkan hasil *clustering* yang dihasilkan dari titik potong yang lain. Pada titik potong 94 dengan jumlah *cluster* yang dihasilkan sebanyak 6 *cluster* menghasilkan nilai *silhouette coefficient* terendah yang artinya seluruh dokumen tidak dikelompokkan pada *cluster* yang tepat pada titik potong tersebut.

6.2 Pengujian Berdasarkan Titik Potong Pada *Complete Linkage*

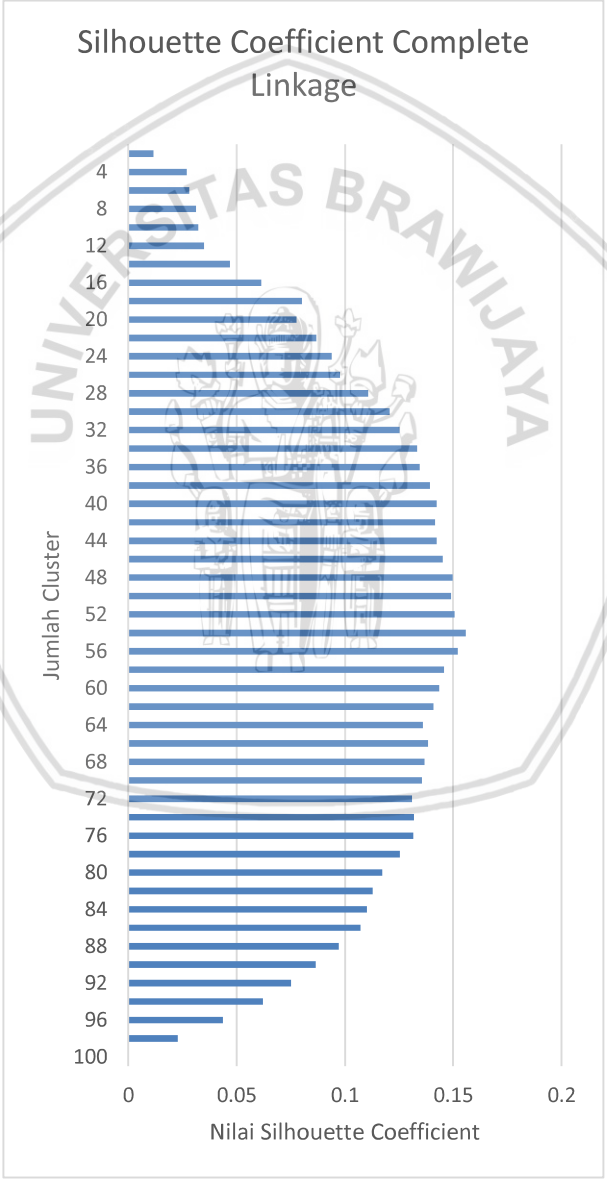
Tabel 6.2 menunjukkan hasil pengujian nilai *Silhouette Coefficient* dari parameter jarak *complete linkage* yang dihasilkan berdasarkan titik potong yang menghasilkan jumlah *cluster* mulai dari 100 *cluster* sampai dengan 2 *cluster*. Setiap titik potong yang diambil bertambah 2 sehingga jumlah *cluster* yang dihasilkan berkurang 2 setiap titik potongnya.

Tabel 6.2 Pengujian berdasarkan titik potong tiap tahap dengan *complete linkage*

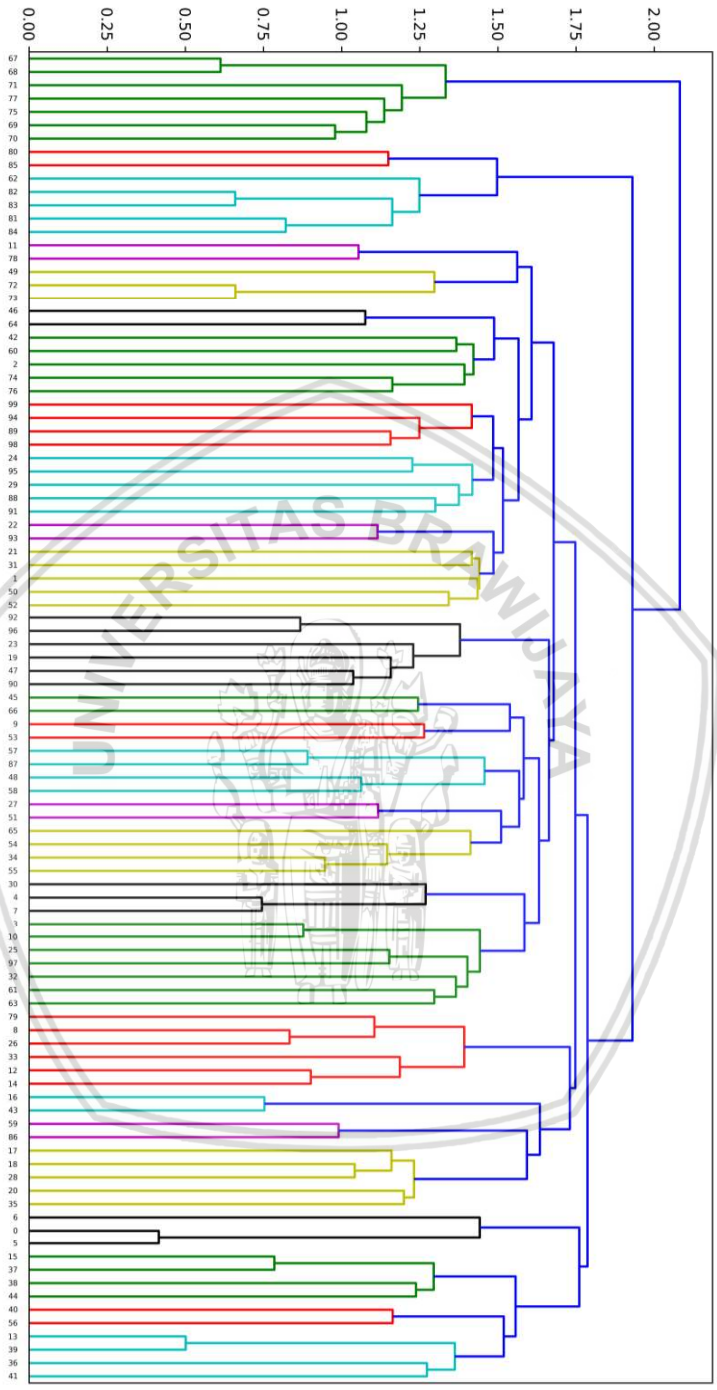
Complete Linkage			*Complete Linkage*		
Titik Potong	Jumlah *Cluster*	SC	Titik Potong	Jumlah *Cluster*	SC
0	100	0	50	50	0,149008
2	98	0,022971	52	48	0,149738
4	96	0,043613	54	46	0,14516
6	94	0,062094	56	44	0,142247
8	92	0,075112	58	42	0,141602
10	90	0,086518	60	40	0,142259
12	88	0,097069	62	38	0,139165
14	86	0,107104	64	36	0,134462
16	84	0,110251	66	34	0,133387
18	82	0,112827	68	32	0,125141
20	80	0,117305	70	30	0,120678
22	78	0,125292	72	28	0,11078
24	76	0,131624	74	26	0,097637
26	74	0,131851	76	24	0,093866
28	72	0,131017	78	22	0,086871
30	70	0,135582	80	20	0,077662
32	68	0,136713	82	18	0,080193
34	66	0,138403	84	16	0,061443
36	64	0,13603	86	14	0,046908
38	62	0,140887	88	12	0,035068
40	60	0,143579	90	10	0,032364

Tabel 6.2 Pengujian berdasarkan titik potong tiap tahap dengan *complete linkage* (lanjutan)

Complete Linkage			*Complete Linkage*		
Titik Potong	Jumlah *Cluster*	SC	Titik Potong	Jumlah *Cluster*	SC
42	58	0,145781	92	8	0,031265
44	56	0,15209	94	6	0,028225
46	54	0,155733	96	4	0,026963
48	52	0,15064	98	2	0,011697



Gambar 6.3 Grafik nilai *Silhouette Coefficient Complete Linkage*



Gambar 6.4 Dendrogram *Complete Linkage*

Gambar 6.4 menunjukkan hasil dendrogram pengelompokan dokumen dengan parameter *complete linkage.*

Bardasarkan Tabel 6.2 dan Gambar 6.2 mengenai hasil pengelompokan dokumen menggunakan parameter jarak *complete linkage* diperoleh nilai *silhouette coefficient* tertinggi pada titik potong 46 dengan jumlah *cluster* sebanyak 54 *cluster.* Nilai rata-rata *silhouette coefficient* dari seluruh dokumen yang dihasilkan pada titik potong 46 yaitu 0,155733. Titik potong 46 memiliki nilai *silhouette coefficient* tertinggi yang artinya seluruh dokumen berada pada *cluster* yang tepat atau dikelompokkan dengan tepat dibandingkan hasil *clustering* yang dihasilkan dari titik potong yang lain. Pada titik potong 98 dengan jumlah *cluster* yang dihasilkan sebanyak 2 *cluster* menghasilkan nilai *silhouette coefficient* terendah yang artinya seluruh dokumen tidak dikelompokkan pada *cluster* yang tepat pada titik potong tersebut.

6.3 Pengujian Berdasarkan Titik Potong Pada *Average Linkage*

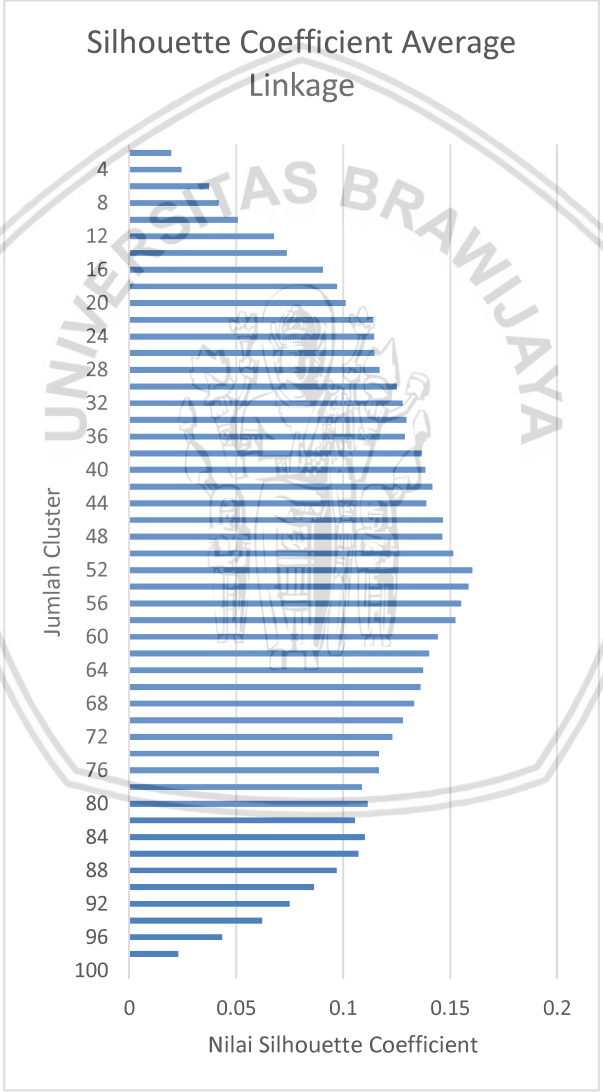
Tabel 6.3 menunjukkan hasil pengujian nilai *Silhouette Coefficient (SC)* dari parameter jarak *average linkage* yang dihasilkan berdasarkan titik potong yang menghasilkan jumlah *cluster* mulai dari 100 *cluster* sampai dengan 2 *cluster*. Setiap titik potong yang diambil bertambah 2 sehingga jumlah *cluster* yang dihasilkan berkurang 2 setiap titik potongnya.

Tabel 6.3 Pengujian berdasarkan titik potong tiap tahap dengan *average linkage*

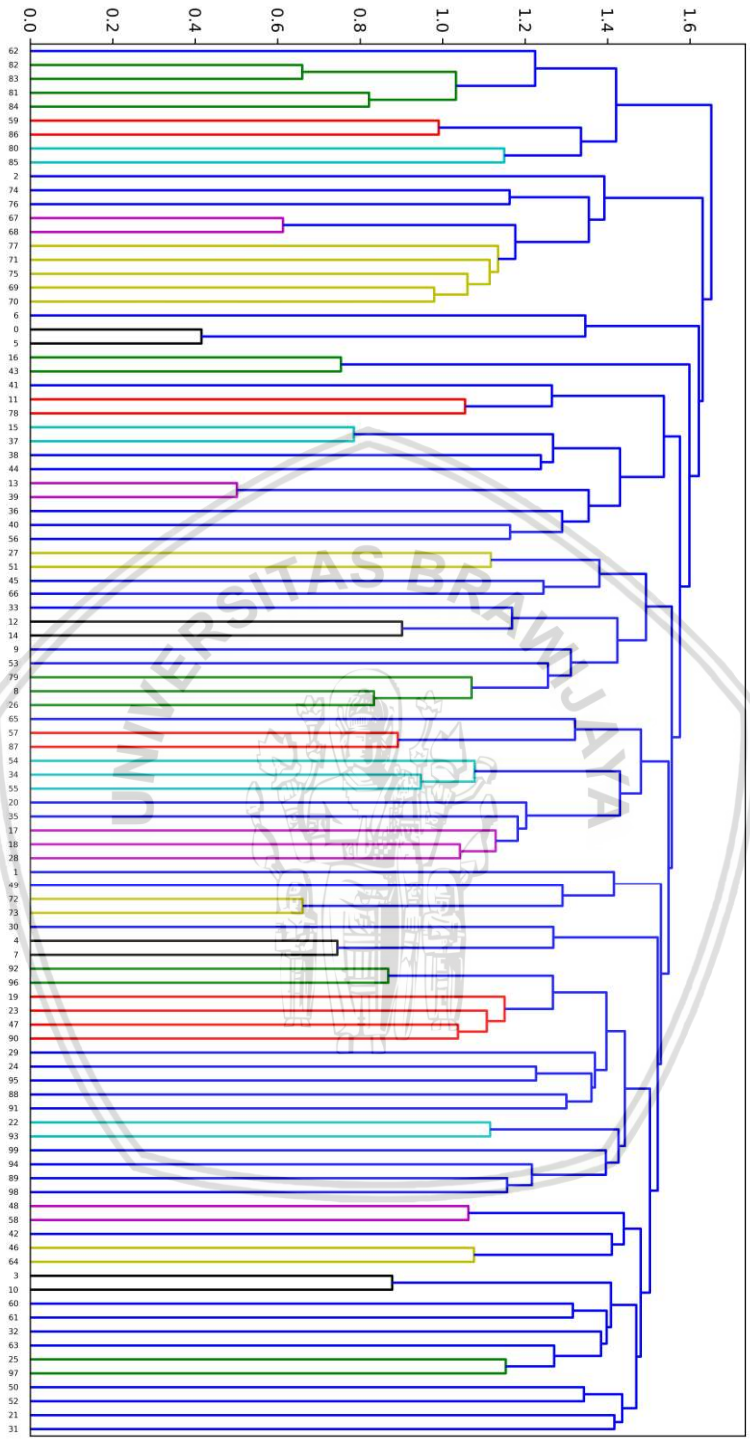
Average Linkage			*Average Linkage*		
Titik Potong	Jumlah *Cluster*	SC	Titik Potong	Jumlah *Cluster*	SC
0	100	0	50	50	0,151441
2	98	0,022971	52	48	0,146349
4	96	0,043613	54	46	0,146704
6	94	0,062094	56	44	0,13895
8	92	0,075112	58	42	0,141697
10	90	0,086518	60	40	0,138476
12	88	0,097069	62	38	0,136909
14	86	0,107104	64	36	0,128877
16	84	0,110251	66	34	0,129667
18	82	0,105595	68	32	0,127808
20	80	0,111482	70	30	0,12513
22	78	0,108793	72	28	0,117129
24	76	0,11678	74	26	0,114624
26	74	0,116745	76	24	0,114468
28	72	0,123077	78	22	0,114111
30	70	0,127973	80	20	0,101278
32	68	0,133172	82	18	0,097213
34	66	0,136266	84	16	0,090506
36	64	0,137464	86	14	0,073688
38	62	0,140256	88	12	0,067773
40	60	0,144355	90	10	0,050881

Tabel 6.3 Pengujian berdasarkan titik potong tiap tahap dengan *average linkage* (lanjutan)

Average Linkage			*Average Linkage*		
Titik Potong	Jumlah *Cluster*	SC	Titik Potong	Jumlah *Cluster*	SC
42	58	0,152629	92	8	0,041891
44	56	0,155306	94	6	0,037349
46	54	0,158619	96	4	0,024437
48	52	0,160428	98	2	0,019841



Gambar 6.5 Grafik nilai *Silhouette Coefficient Average Linkage*



Gambar 6.6 Dendrogram *Average Linkage*

Gambar 6.6 menunjukkan hasil dendrogram pengelompokan dokumen dengan parameter *average linkage.*

Bardasarkan Tabel 6.3 dan Gambar 6.3 mengenai hasil pengelompokan dokumen menggunakan parameter jarak *average linkage* diperoleh nilai *silhouette coefficient* tertinggi pada titik potong 48 dengan jumlah *cluster* sebanyak 52 *cluster.* Nilai rata-rata *silhouette coefficient* dari seluruh dokumen yang dihasilkan pada titik potong 86 yaitu 0,160428. Titik potong 48 memiliki nilai *silhouette coefficient* tertinggi yang artinya seluruh dokumen berada pada *cluster* yang tepat atau dikelompokkan dengan tepat dibandingkan hasil *clustering* yang dihasilkan dari titik potong yang lain. Pada titik potong 98 dengan jumlah *cluster* yang dihasilkan sebanyak 2 *cluster* menghasilkan nilai *silhouette coefficient* terendah yang artinya seluruh dokumen tidak dikelompokkan pada *cluster* yang tepat pada titik potong tersebut.

6.4 Analisis Hasil Pengujian

6.4.1 Analisis Hasil Pengujian *Silhouette Coefficient* Tertinggi Dari Parameter Jarak(*Linkage*)

Hasil dari pengujian 3 parameter jarak (*linkage*) mendapatkan nilai *silhouette coefficient* tertinggi yang berbeda-beda. Setiap nilai *silhouette coefficient* tertinggi dari masing-masing parameter jarak akan dibandingkan untuk dilihat parameter jarak (*linkage*) yang memiliki nilai *silhouette coefficient* tertinggi. Tabel 6.4 menunjukkan berbandingan nilai *silhouette coefficient* tertinggi dari masing-masing parameter jarak (*linkage*).

Tabel 6.4 Nilai *silhouette coefficient* tertinggi dari tiap-tiap parameter jarak (*linkage*)

Parameter Jarak (Linkage)	*Silhouette Coefficient*
Single Linkage	0,10125
Complete Linkage	0,155733
Average Linkage	0,160428

Dari perbandingan nilai *silhouette coefficient* masing-masing parameter jarak dihasilkan nilai *silhouette coefficient* tertinggi adalah pada penggunaan parameter *Average Linkage.* Nilai *silhouette coefficient* tertinggi yang dihasilkan dari *average linkage* yaitu 0,160428.

Nilai *silhouette coefficient* memiliki jangkauan antara -1 sampai dengan 1 (Rousseeuw, 1987). Jika nilai *silhouette coefficient* sama dengan 1 atau semakin mendekati 1 maka setiap dokumen berada pada *cluster* yang tepat. Jika nilai *Silhouette Coefficient* sama dengan 0 atau lebih mendekati 0 maka setiap dokumen tidak terlalu tepat berada dalam suatu *cluster* atau suatu dokumen masih dapat masuk ke *cluster* lain. Jika nilai *Silhouette Coefficient* sama dengan -1 atau mendekati -1 maka dokumen-dokumen yang terkelompokkan pada *cluster* tersebut tidak seharusnya berada *cluster* itu namun lebih tepat untuk masuk ke *cluster* lain.

Nilai *silhouette coefficient* yang dihasilkan dari pengujian lebih mendekati angka 0, Dari hasil pengujian tersebut maka *cluster-cluster* yang dihasilkan tidak terlalu kuat atau dokumen-dokumen masih dapat masuk ke *cluster* lain. Dari hasil pengujian juga dapat disimpulkan jika semakin banyak *Singleton* (*cluster* dengan jumlah 1 dokumen) maka semakin rendah nilai *Silhouette Coefficient* yang dihasilkan namun apabila jumlah *cluster* yang dihasilkan semakin sedikit maka semakin rendah pula nilai *silhouette coefficient*. Tinggi atau rendahnya nilai *silhouette coefficient* ditentukan oleh jarak atau nilai *cosine distance* dari suatu dokumen ke dokumen lainnya. Nilai *cosine distance* diperoleh dari nilai *cosine similarity* sehingga apabila dokumen-dokumen yang dipilih sebagai dataset tidak terlalu memiliki kemiripan satu sama lainnya atau tingkat kemiripannya rendah maka nilai *silhouette coefficient* yang dihasilkan juga rendah.

Tingkat kekuatan yang dihasilkan dari pengujian dengan *Silhouette Coefficient* dapat dipengaruhi oleh beberapa hal yaitu dokumen yang digunakan, jumlah dokumen, jumlah *cluster* yang dihasilkan, jumlah *singleton* yang ada.

6.4.2 Analisis Hasil Pengujian *Silhouette Coefficient* Tiap Dokumen

Tabel 6.5 menunjukkan nilai *silhouette coefficient* tiap dokumen pada titik potong 48 pada *average linkage* yang mana pada parameter tersebut menghasilkan nilai rata-rata *silhouette coefficient* tertinggi.

Tabel 6.5 Nilai *silhouette coefficient* tiap dokumen pada parameter dengan rata-rata *silhouette coefficient* tertinggi

SC Tiap Dokumen			SC Tiap Dokumen		
Cluster	**Dokumen**	**Silhouette Coefficient**	***Cluster***	**Dokumen**	**Silhouette Coefficient**
0	1	0	36	54	0,208579
1	2	0	37	27	0,169859
2	6	0	37	51	0,149745
3	9	0	38	22	0,177209
4	21	0	38	93	0,164916
5	24	0	39	11	0,242638
6	29	0	39	78	-0,04203
7	30	0	39	41	0,083797
8	31	0	40	82	0,202535
9	32	0	40	83	0,295934
10	36	0	40	81	0,19724
11	42	0	40	84	0,27453
12	44	0	40	62	0,140966
13	49	0	41	12	0,223925
14	50	0	41	14	0,204375
15	52	0	41	33	0,127273
16	60	0	42	63	0,125752
17	61	0	42	97	0,056649
18	65	0	43	74	0,078384

Tabel 6.5 Nilai *silhouette coefficient* tiap dokumen pada parameter dengan rata-rata *silhouette coefficient* tertinggi (lanjutan)

SC Tiap Dokumen			SC Tiap Dokumen		
Cluster	**Dokumen**	**Silhouette Coefficient**	***Cluster***	**Dokumen**	**Silhouette Coefficient**
19	85	0	43	76	0,090317
20	88	0	44	23	0,088221
21	91	0	44	90	0,127911
22	95	0	44	47	0,124504
23	99	0	44	19	0,101863
24	0	0,75971	45	67	0,15231
24	5	0,681049	45	68	0,268129
25	13	0,595703	45	75	0,136107
25	39	0,58693	45	69	0,182003
26	16	0,513083	45	70	0,199924
26	43	0,521435	45	71	0,116292
27	72	0,466006	45	77	0,140016
27	73	0,470406	46	45	0,021173
28	4	0,404969	46	66	0,077131
28	7	0,403128	47	25	0,066085
29	15	0,36369	47	38	0,013814
29	37	0,354953	48	17	0,135335
30	3	0,321722	48	18	0,116979
30	10	0,341896	48	35	0,122986
31	92	0,350166	48	20	0,103209
31	96	0,273448	49	8	0,113888
32	57	0,24373	49	26	0,170442
32	87	0,359208	49	79	0,123989
33	40	0,10576	49	53	0,051767
33	56	0,119514	50	28	-0,08397
34	48	0,230369	50	59	0,230709
34	58	0,227615	50	80	-0,08287
35	46	0,218867	50	86	0,158518
35	64	0,166589	51	89	0,133461
36	34	0,27811	51	98	0,071924
36	55	0,237752	51	94	0,09252

Dari Tabel 6.5 dokumen 0 pada *cluster* 24 memiliki nilai *silhouette coefficient* tertinggi yaitu 0,75971 yang artinya dokumen tersebut tepat dikelompokkan dengan dokumen yang berada pada *cluster* 24, sedangkan dokumen 28 pada *cluster* 50 memiliki nilai *silhouette coefficient* terendah yaitu -0,08397 yang artinya dokumen tersebut tidak tepat dikelompokkan bersama dokumen-dokumen pada *cluster* 50,

6.4.3 Analisis Hasil Pengujian *Silhouette Coefficient* Tiap *Cluster*

Tabel 6.6 menunjukkan nilai *silhouette coefficient* tiap *cluster* berdasarkan titik potong 48 pada *average linkage.* Nilai *silhouette coefficient* tiap *cluster* diperoleh dari rata-rata *silhouette coefficient* seluruh dokumen pada masing-masing *cluster.*

Tabel 6.6 Nilai *silhouette coefficient* tiap *cluster* pada parameter dengan rata-rata *silhouette coefficient* tertinggi

SC Tiap *Cluster*			SC Tiap *Cluster*		
Cluster	**Dokumen**	**Silhouette Coefficient**	***Cluster***	**Dokumen**	**Silhouette Coefficient**
0	1	0	26	[16, 43]	0,517259
1	2	0	27	[72, 73]	0,468206
2	6	0	28	[4, 7]	0,404048
3	9	0	29	[15, 37]	0,359322
4	21	0	30	[3, 10]	0,331809
5	24	0	31	[92, 96]	0,311807
6	29	0	32	[57, 87]	0,301469
7	30	0	33	[40, 56]	0,112637
8	31	0	34	[48, 58]	0,228992
9	32	0	35	[46, 64]	0,192728
10	36	0	36	[34, 55, 54]	0,24148
11	42	0	37	[27, 51]	0,159802
12	44	0	38	[22, 93]	0,171062
13	49	0	39	[11, 78, 41]	0,094803
14	50	0	40	[82, 83, 81, 84, 62]	0,222241
15	52	0	41	[12, 14, 33]	0,185191
16	60	0	42	[63, 97]	0,0912
17	61	0	43	[74, 76]	0,084351
18	65	0	44	[23, 90, 47, 19]	0,110625
19	85	0	45	[67, 68, 75, 69, 70, 71, 77]	0,170683
20	88	0	46	[45, 66]	0,049152
21	91	0	47	[25, 38]	0,03995
22	95	0	48	[17, 18, 35, 20]	0,119627
23	99	0	49	[8, 26, 79, 53]	0,115021
24	[0, 5]	0,72038	50	[28, 59, 80, 86]	0,055597
25	[13, 39]	0,591317	51	[89, 98, 94]	0,099302

Dari Tabel 6.6 diapat dilihat *cluster* 24 yang terdiri dari dokumen 0 dan dokumen 5 meiliki nilai *silhouette coefficient* tertinggi yaitu 0,72038 yang artinya *cluster* tersebut kuat karena dokumen-dokumen didalamnya memiliki kemiripan yang tinggi dan berbeda dengan dokumen dari *cluster* lain.

6.4.4 Hasil Pelabelan *Cluster*

Tabel 6.7 menunjukkan hasil pelabelan dari *cluster* yang diambil berdasarkan nilai *silhouette coefficient* tertinggi. *Cluster* yang diberi label adalah *cluster* yang dihasilkan dari parameter jarak (*linkage*) dengan rata-rata *silhouette coefficient* tertinggi yaitu pada *average linkage* dengan titik potong 48. *Cluster* yang dihasilkan sebanyak 52 *cluster.* Label-label dari *cluster* didapatkan dari nilai bobot *TF-IDF* tiap *cluster. Term-term* dari *cluster singleton* sama dengan *term-term* yang dihasilkan pada 1 dokumen yang ada pada *cluster* tersebut. *Term* pada *cluster* selain *singleton* merupakan *term* gabungan seluruh dokumen yang terdapat pada *cluster* tersebut.

Tabel 6.7 Hasil pelabelan *cluster* pada parameter dengan rata-rata *silhouette coefficient* tertinggi

Cluster Terbentuk			
Cluster	**Dokumen**	**Label**	**Judul Skripsi**
0	1	['chain', 'toys', 'factory', 'baiducha', 'technology']	1: Analisis dan Implementasi Load Balancing pada Web Server dengan Algoritma Least Connection.
1	2	['2', 'smkn', 'enterprise', 'planning', 'resource']	2: Analisis Dan Pemodelan Sistem Pengadaan Barang Dengan Supply Chain Management Pada Pabrik Toys Factory Baiducha Technology
2	6	['bellman-ford', 'banding', 'dijkstra', 'rute', 'pendek']	6: Analisis Perbandingan Algoritma Dijkstra Dan Bellman-Ford Untuk Menentukan Rute Terpendek Dengan Menggunakan Software Defined Network
3	9	['media', 'elite', 'lalu', 'kernel', 'politik']	9: Analisis Sentimen Pencitraan Elite Politik Berdasarkan Opini Melalui Media Sosial Twitter Menggunakan Metode Additive Kernel SVM
4	21	['rprop', 'resilient', 'keluarga', 'ajar', 'tiru']	21: Implementasi Jaringan Syaraf Tiruan Dengan Metode Pembelajaran Rprop (Resilient Backpropagation) Untuk Klasifikasi Status Tahapan Keluarga Sejahtera
5	24	['sumber', 'manusia', 'organisasi', 'daya', 'panitia']	24: Implementasi Metode Fuzzy Analytical Hierarchy Process (F-Ahp) Untuk Pemilihan Sumber Daya Manusia Dalam Kepanitiaan Organisasi Mahasiswa
6	29	['karyawan', 'making', 'attribute', 'sierad', 'produce']	29: Implementasi Metode Topsis - Multiple Attribute Decision Making Untuk Pemilihan Karyawan Berprestasi Berdasarkan Kinerja (Studi Kasus Pada PT Sierad Produce, tbk)
7	30	['perangkat', 'hoc',	30: Implementasi Mobile Ad Hoc Network (MANET) Menggunakan Protokol Routing OLSR Pada Perangkat Heterogen

		'heterogen', 'manet', 'olsr']	
8	31	['pathfinding', 'real-time', 'reynolds', 'main', 'steering']	31: Implementasi Real-time Pathfinding menggunakan A* dan Reynolds Steering Obstacle Avoidance pada Permainan Komputer
9	32	['point', 'private', 'mikrotik', 'ukm', 'radio']	32: Implementasi Wifi Network dan Virtual Private Network dengan menggunakan Radio Point to Point Berbasis Mikrotik pada UKM Menwa Universitas Brawijaya,
10	36	['ibu', 'hamil', 'asupan', 'gizi', 'genetika']	36: Optimasi Asupan Gizi Pada Ibu Hamil Dengan Menggunakan Algoritma Genetika
11	42	['pbb', 'jombang', 'pajak', 'framework', 'codeigniter']	42: Pelaporan Dan Pengolahan Data Pajak Bumi Dan Bangunan (PBB) Kabupaten Jombang Berbasis Web Menggunakan Framework CodeIgniter.
12	44	['inference', 'metabolisme', 'angka', 'system', 'basal']	44: Penerapan Algoritma Genetika Untuk Optimasi Fungsi Keanggotaan Fuzzy Inference System Model Sugeno Pada Perhitungan Angka Metabolisme Basal (AMB).
13	49	['company', 'kembang', 'clothing', 'clove', 'supplier']	49: Pengembangan Sistem Informasi Supplier Relationship Management (SRM) Pada Clove Clothing Company
14	50	['abadi', 'pasang', 'mode', 'gagal', 'baik']	50: Penggunaan Fuzzy Failure Mode And Effect Analysis (Fuzzy Fmea) Dalam Mengidentifikasi Resiko Kegagalan Proses Pemasangan Dan Perbaikan Ac (Di Cv. Agung Jaya Abadi)
15	52	['testing', 'tiga', 'prototype', 'putra', 'usability']	52: Penggunaan Usability Testing Untuk Menguji Prototype Crm Pada Cv Tiga Putra
16	60	['code', 'inventaris', 'qr', 'ptiik', 'informasi']	60: Perancangan dan Analisis Data Sistem Informasi Inventaris PTIIK dengan Menggunakan QR Code.
17	61	['sedia', 'arima', 'pustaka', 'subjek', 'buku']	61: Perancangan Dan Analisis Sistem Peramalan Dan Rekomendasi Penyediaan Subjek Buku Pada Perpustakaan Universitas Brawijaya Menggunakan Metode Arima
18	65	['minta', 'jumlah', 'koran',	65: Prediksi Jumlah Permintaan Koran Menggunakan Metode Extreme Learning Machine

		'extreme', 'prediksi']	
19	85	['cegah', 'faktor', 'serviks', 'kanker', 'resiko']	85: Sistem Pakar Pencegahan Dini terhadap Kanker Serviks Berdasarkan Faktor Resiko Menggunakan Metode Certainty Factor Berbasis Mobile Web
20	88	['transportasi', 'floyd-warshall', 'umum', 'alat', 'jalur']	88: Sistem Pendukung Keputusan Pemilihan Alat Transportasi Umum Kota Malang Berdasar Jalur Terpendek dengan Menggunakan Algoritma Floyd-Warshall
21	91	['promethee', 'psikologi', 'layan', 'umm', 'plp']	91: Sistem Pendukung Keputusan Pemilihan Calon Pegawai Marketing Dengan Menggunakan Metode Promethee (Studi Kasus Pusat Layanan Psikologi (Plp) Umm Malang)
22	95	['didik', 'uptd', 'cabang', 'dinas', 'sertifikasi']	95: Sistem Pendukung Keputusan Penetapan Calon Peserta Sertifikasi Guru Sekolah Dasar Menggunakan Metode Analytical Hierarchy Process (AHP) (Studi Kasus : Uptd Cabang Dinas Pendidikan Buduran)
23	99	['miskin', 'tangga', 'rumah', 'product', 'proses']	99: Sistem Pendukung Keputusan untuk Proses Penentuan Rumah Tangga Miskin Menggunakan Metode Weighted Product.
24	[0, 5]	['balancing', 'least', 'connection', 'load', 'server']	0: Analisis dan Implementasi Load Balancing pada Web Server dengan Algoritma Least Connection.
			5: Analisis Openflow Load Balancing Web Server Dengan Algoritma Least Connection Pada Software Defined Network
25	[13, 39]	['komposisi', 'derita', 'makan', 'mellitus', 'diabetes']	13: Implementasi Algoritma Genetika Untuk Optimasi Komposisi Makanan Bagi Penderita Diabetes Mellitus.
			39: Optimasi Komposisi Makanan Untuk Penderita Kolesterol Menggunakan Algoritma Genetika
26	[16, 43]	['subtractive', 'bangkit', 'atur', '*cluster*ing', 'beasiswa']	16: Implementasi Algoritma Subtractive *Cluster*ing Untuk Pembangkitan Aturan Fuzzy Pada Rekomendasi Penerima Beasiswa
			43: Pembangkitan Aturan Fuzzy Menggunakan Metode Subtractive *Cluster*ing Untuk Deteksi Dini Risiko Penyakit Stroke

27	[72, 73]	['customer', 'toko', 'relationship', 'buku', 'electronic']	72: Rancang Bangun Sistem Customer Relationship Management (CRM) Pada Toko Buku Qudsi Malang Menggunakan Metode Prototyping
			73: Rancang Bangun Sistem E-CRM (Electronic Customer Relationship Management) (Studi Kasus : Toko Buku Togamas Malang).
28	[4, 7]	['path', 'first', 'shortest', 'open', 'routing']	4: Analisis Kinerja Protokol Routing Open Shortest Path First Pada Teknologi Software-Defined Networking
			7: Analisis Protokol routing OSPF (Open Shortest Path First) pada Wireless Mesh Network dalam kasus traffic data Voice
29	[15, 37]	['swarm', 'optimization', 'particle', 'inferensifuzzy', 'jurus']	15: Implementasi Algoritma Particle Swarm Optimization Untuk Optimasi Fungsi Keanggotaan Sistem Inferensifuzzy Mamdanipada Penyakit Hepatitis
			37: Optimasi Fungsi Keanggotaan Fuzzy Menggunakan Algoritma Particle Swarm Optimization (Pso) Pada Sistem Inferensi Fuzzy Penentuan Jurusan Siswa SMA
30	[3, 10]	['evaluation', 'heuristic', 'usability', 'situs', 'evaluasi']	3: Analisis Implementasi persona Pada Penerapan Metode Evaluasi Usability Heuristic Evaluation. Studi Kasus : Situs Web Fakultas Ilmu Komputer Universitas Brawijaya (Filkom UB)
			10: Analisis Usability Pada Website Universitas Brawijaya Dengan Heuristic Evaluation
31	[92, 96]	['similarity', 'technique', 'ideal', 'preference', 'for']	92: Sistem Pendukung Keputusan Pemilihan Simplisia Nabati Terhadap Indikasi Gangguan Kesehatan Menggunakan Metode Analytic Hierarchy Process - The Technique for Order of Preference by Similarity to Ideal
			96: Sistem Pendukung Keputusan Rekomendasi Peringkat Asuransi Kesehatan Menggunakan Metode Analytic Network Process (ANP) dan Technique For Others Preference By Similarity Ideal Solution (TOPSIS) Studi Ka
32	[57, 87]	['gula', 'sidoarjo', 'candi', 'pg', 'baru']	57: Peramalan Produksi Gula Pasir Menggunakan Extreme Learning Machine (ELM) pada PG Candi Baru Sidoarjo

			87: Sistem Pelaporan Dan Peramalan Penjualan Gula Dengan Mengimplementasikan Metode Exponential Smoothing Pada eknologi Rolap (Studi Kasus pada PT PG Candi Baru Sidoarjo)
33	[40, 56]	['jadwal', 'semester', 'awas', 'awat', 'fakultas']	40: Optimasi Penjadwalan Perawat Menggunakan Algoritma Genetika
			56: Penjadwalan Pengawas Ujian Semester Menggunakan Algoritma Genetika (Studi Kasus : Fakultas Ilmu Komputer Universitas Brawijaya)
34	[48, 58]	['series', 'time', 'tampung', 'hari', 'saham']	48: Penerapan Metode Fuzzy Time Series Average-Based Pada Peramalan Data Harian Penampungan Susu Sapi
			58: Peramalan Time Series Saham Menggunakan Backrpopagaion Neural Network Berbasis Algoritma Genetika
35	[46, 64]	['tolerant', 'delay', 'konsultasi', 'dtn', 'tukar']	46: Penerapan Delay Tolerant Network (DTN) untuk Sistem Konsultasi Kesehatan Jarak Jauh Berbasis Web,
			64: Perancangan Sistem Pertukaran Informasi Di Pedesaan Berbasis Delay Tolerant Network menggunakan Raspberry PI.
36	[34, 55, 54]	['quantization', 'lvq', 'vector', 'learning', 'gigi']	34: Klasifikasi Penyakit Gigi dan Mulut Menggunakan Metode Learning Vector Quantization (Lvq)
			55: Pengklasifikasian Mutu Susu Sapi Menggunakan Metode Learning Vector Quantization (LVQ) (Studi Kasus: UPT Laboratorium Kesehatan Hewan Malang)
			54: Pengklasifikasian Kualitas Minuman Anggur Menggunakan Algoritma Learning Vector Quantization Berbasis Asosiasi
37	[27, 51]	['k-means', 'kanker', 'protein', 'jenis', 'susun']	27: Implementasi Metode Klustering Untuk Klasifikasi Kanker Payudara Menggunakan Algoritma Naïve Bayes Dan K-Means
			51: Penggunaan Metode Pengelompokan K-Means Pada Klasifikasi KNN Untuk Penentuan Jenis Kanker Berdasarkan Susunan Protein
38	[22, 93]	['ayam', 'ahp', 'broiler', 'isi', 'telur']	22: Implementasi Metode AHP – Fuzzy TOPSIS Untuk Rekomendasi Penentuan Tingkat Kualitas Produktivitas Ayam Ras Petelur

			93: Sistem Pendukung Keputusan Penentuan Kelayakan Pengisian Bibit Ayam Broiler Dikandang Peternak Menggunakan Metode AHP dan TOPSIS
39	[11, 78, 41]	['kuliner', 'rute', 'kota', 'lokasi', 'angkut']	11: Aplikasi Penentuan Rute Lokasi Kuliner Di Kota Malang Berbasis Gis Menggunakan Metode A*
			78: Rancang Bangun Sistem Rekomendasi Kuliner Kota Malang Dengan Berbasis Web Menggunakan Metode Weighted Product
			41: Optimasi Rute Angkutan Kota Malang Dengan Penerapan Algoritma Genetika
40	[82, 83, 81, 84, 62]	['diagnosa', 'anak', 'naive', 'pakar', 'kulit']	82: Sistem Pakar Diagnosa Penyakit Kulit Pada Anak Menggunakan Metode Certainty Factor.
			83: Sistem Pakar Diagnosa Penyakit Kulit pada Anak Menggunakan Metode Naïve Bayes
			81: Sistem Pakar Diagnosa Penyakit Demam Berdarah Menggunakan Metode Naive Bayes dan Certainty Factor.
			84: Sistem Pakar Diagnosa Penyakit Sapi Potong dengan Metode Naive Bayes.
			62: Perancangan dan Implementasi Sistem Pakar Pendukung Diagnosa Penyakit Anjing dengan Metode Bayesian Network.
41	[12, 14, 33]	['modified', 'neighbor', 'tanam', 'diagnosa', 'k-nearest']	12: Diagnosa Penyakit Tanaman Cabai Merah Menggunakan Metode Modified K-Nearest Neighbor (MK-NN)
			14: Implementasi Algoritma Modified K-Nearest Neighbor (MKNN) untuk Klasifikasi Diagnosa Penyakit pada Kucing
			33: Klasifikasi Dokumen Tanaman Obat Menggunakan Metode Neighbor Weighted K-Nearest Neighbor (NWKNN).
42	[63, 97]	['informatika', 'teknik', 'video', 'lan', 'profile']	63: Perancangan Dan Implementasi Video Streaming Client Android Dalam Jaringan Wireless LAN Di Teknik Informatika Universitas Brawijaya.
			97: Sistem Pendukung Keputusan Seleksi Penerimaan Asisten Praktikum Menggunakan Metode Profile Matching (Studi Kasus Prodi Teknik Informatika Universitas Brawijaya)
43	[74, 76]	['manajemen', 'program',	74: Rancang Bangun Sistem Informasi Manajemen Kepegawaian untuk Penggajian

		'gaji', 'unjung', 'taraperkasa']	pada PT. Gandapahala Taraperkasa berbasis aplikasi WEB.
			76: Rancang Bangun Sistem Manajemen Pengunjung Laboratorium Program Teknologi Informasi dan Ilmu Komputer (PTIIK) Berbasis Pengenalan Wajah.
44	[23, 90, 47, 19]	['anp', 'analytic', 'guru', 'process', 'maospati']	23: Implementasi Metode Analytic Network Process (Anp) Untuk Aplikasi Rekomendasi Peringkat Kinerja Guru Pada Sma Negeri 1 Maospati
			90: Sistem Pendukung Keputusan Pemilihan Bidang Studi di Perguruan Tinggi Menggunakan Metode Analytic Network Process (ANP).
			47: Penerapan Metode Analytic Network Process (Anp) Sebagai Sistem Pengambil Keputusan Untuk Aplikasi Pemilihan Penginapan Di Kota Batu
			19: Implementasi Analytic Network Process (ANP) Dalam Sistem Pendukung Keputusan Pemilihan Kontraktor (StudiKasus PT. Pelabuhan Indonesia III (Persero) CabangTanjung Perak)
45	[67, 68, 75, 69, 70, 71, 77]	['smartphone', 'android', 'gedung', 'rancang', 'bangun']	67: Rancang Bangun Aplikasi Informasi Gedung Di Universitas Brawijaya Malang Pada Posisi Pengguna Dengan Sistem Operasi Android
			68: Rancang Bangun Aplikasi Informasi Gedung universitas Brawijaya Pada Smartphone Android
			75: Rancang Bangun Sistem Informasi Pariwisata Berbasis Augmented Reality pada Smartphone Android.
			69: Rancang Bangun Aplikasi Jejaring Sosial Kampus Berbasis GPS Pada Smartphone Android.
			70: Rancang Bangun Aplikasi Simulasi Gadai Pada Smartphone Android
			71: Rancang Bangun E-Learning Berbasis Moodle Menggunakan Aplikasi Mobile Android
			77: Rancang Bangun Sistem Navigasi Antar Smartphone Berbasis Android dengan Google App Engine.
46	[45, 66]	['jantung', 'pjk', 'fk-nn',	45: Penerapan Algoritma Improved K-Means Pada Pengelompokan Data Tingkat Kesehatan Jantung.

		'koroner', 'tingkat']	
			66: Prediksi Tingkat Risiko Penyakit Jantung Koroner (PJK) Menggunakan Metode Fuzzy K-Nearest Neighbor (FK-NN)
47	[25, 38]	['suara', 'padu', 'bbp-ppa', 'beasiswa-ppa', 'tsukamoto']	25: Implementasi Metode Fuzzy-Ahp Untuk Rekomendasi Seleksi Penerimaan Anggota Baru Paduan Suara (Studi Kasus: Paduan Suara Mahasiswa Universitas Brawijaya)
			38: Optimasi Fungsi Keanggotaan Fuzzy Tsukamoto Dua Tahap Menggunakan Algoritma Genetika Pada Pemilihan Calon Penerima Beasiswa-PPA dan BBP-PPA (Studi Kasus: PTIIK Universitas Brawijaya Malang)
48	[17, 18, 35, 20]	['support', 'jalan', 'rel', 'vector', 'svm']	17: Implementasi algoritma support vector machine (svm) untuk penentuan potensi bencana tsunami akibat gempa bumi
			18: Implementasi Algoritma Svm (Support Vector Machine) Untuk Mengetahui Tingkat Risiko Penyakit Stroke
			35: Klasifikasi Tingkat Kerusakan Jalan Rel Kereta Api Menggunakan Metode Support Vector Machine (Svm) (Studi Kasus : Jalan Rel 8.16 Malang)
			20: Implementasi Fuzzy Support Vector Machine Untuk Pengklasifikasian Genre Musik Berdasarkan Fitur Audio.
49	[8, 26, 79, 53]	['bahasa', 'indonesia', 'sentimen', 'twitter', 'neighbor']	8: Analisis Sentimen Opini Film Berbahasa Indonesia Berbasis Kamus Menggunakan Metode Neighbor- Weigthed K-Nearest Neighbor
			26: Implementasi Metode Improved K-Nearest Neighbor pada Analisis Sentimen Twitter Berbahasa Indonesia.
			79: Sentiment Analysis pada Review Barang Berbahasa Indonesia dengan Metode K-Nearest Neighbor (K-NN).
			53: Pengkategorian Pesan Singkat Berbahasa Indonesia Pada Jejaring Sosial Twitter Dengan Metode Klasifikasinaïve Bayes
50	[28, 59, 80, 86]	['hama', 'digital', 'deteksi', 'tanam', 'citra']	28: Implementasi Metode Support Vector Machine (SVM) Untuk Deteksi Penyakit Pada Daun Apel Menggunakan Citra Digital
			59: Perancangan Aplikasi Pengolahan Citra Digital Untuk Mendeteksi Hama Dan Penyakit Pada Tanaman Jagung

			80: Sistem Pakar Berbasis Web Untuk Identifikasi Hama Penyakit Pada Budidaya Tanaman Jamur Menggunakan Metode Certainty Factor
			86: Sistem Pakar Pendeteksi Hama dan Penyakit Tanaman Mangga dengan Metode Demspter-Shafer.
51	[89, 98, 94]	['weighting', 'simple', 'additive', 'fuzzy-simple', 'mandiri']	89: Sistem Pendukung Keputusan Pemilihan Atlet yang Layak Masuk Tim Pencak Silat Dengan Metode Simple Additive Weighting (SAW).
			98: Sistem Pendukung Keputusan Seleksi Penerimaan Pegawai MKS (Mikro Kredit Sales) Menggunakan Fuzzy Simple Additive Weighting (Studi Kasus: Bank Mandiri Cab. Tulungagung).
			94: Sistem Pendukung Keputusan Penentuan Obat Perawatan Kulit Wajah Menggunakan Metode Fuzzy-Simple Additive Weighting (F-SAW).

Pelabelan *cluster* menggunakan pembobotan *TF-IDF* menghasilkan kandidat label dari *term* yang unik dari tiap-tiap *cluster.* Bobot Tf untuk mendapatkan *term* dengan frekuensi tertinggi, sedangkan *idf* untuk mendapatkan nilai dari suatu *term* apakah sering muncul diseluruh *cluster* atau tidak. Kandidat label yang muncul dari *cluster* adalah *term* yang sering muncul pada suatu *cluster* namun tidak banyak muncul pada *cluster* lain.

6.4.5 Analisis Hasil *Precision* Dari Pelabelan *Cluster*

Tabel 6.8 menunjukkan hasil precision dari pelabelan *cluster* yang diambil berdasarkan nilai *silhouette coefficient* tertinggi. *Cluster* yang diberi label adalah *cluster* yang dihasilkan dari parameter jarak (*linkage*) dengan rata-rata *silhouette coefficient* tertinggi yaitu pada *average linkage* dengan titik potong 48. *Cluster* yang dihasilkan sebanyak 52 *cluster.* Nilai *precision* diperoleh dari label yang diperoleh yang relevan dengan kata kunci dibagi dengan keseluruhan label yang diperoleh.

Tabel 6.8 Nilai *Precision* Dari Hasil Pelabelan *Cluster*

Cluster	Label	Keyword	Precision
0	['chain', 'toys', 'factory', 'baiducha', 'technology']	['ada', 'barang', 'supply', 'chain', 'management', 'black-box', 'testing']	0,2
1	['2', 'smkn', 'enterprise', 'planning', 'resource']	['enterprise', 'resource', 'planning', 'erp', 'enterprise', 'resource', 'planning', 'education', 'erpe', 'prototyping', 'smkn', 'enterprise', 'resource', 'planning', 'erp',	0,8

		'enterprise', 'resource', 'planning', 'education', 'erpe', 'prototyping', 'smk']	
2	['bellman-ford', 'banding', 'dijkstra', 'rute', 'pendek']	['software', 'defined', 'network', 'dijkstra', 'bellman-ford', 'pyretic']	0,4
3	['media', 'elite', 'lalu', 'kernel', 'politik']	['analisis', 'sentimen', 'citra', 'tweets', 'additive', 'kernel', 'svm', 'sentiment', 'analysis', 'imaging', 'tweets', 'additive', 'kernel', 'svm']	0,2
4	['rprop', 'resilient', 'keluarga', 'ajar', 'tiru']	['keluarga', 'sejahtera', 'jaring', 'syaraf', 'tiru', 'resilient', 'backpropagation']	0,6
5	['sumber', 'manusia', 'organisasi', 'daya', 'panitia']	['sdm', 'organisasi', 'pilih', 'panitia', 'mcdm', 'fuzzy', 'ahp']	0,4
6	['karyawan', 'making', 'attribute', 'sierad', 'produce']	['entropy', 'topsis', 'sensitivitas', 'karyawan', 'prestasi']	0,2
7	['perangkat', 'hoc', 'heterogen', 'manet', 'olsr']	['wireless', 'mobile', 'ad', 'hoc', 'network', 'manet', 'olsr', 'self-configure', 'self-healing']	0,6
8	['pathfinding', 'real-time', 'reynolds', 'main', 'steering']	['pathfinding', 'real-time', 'pathfinding', 'a', '', 'algorithm', 'reynolds', 'steering', 'computer', 'games', 'cari', 'jalur', 'real-time', 'pathfinding', 'algoritma', 'a', 'reynolds', 'steering', 'main', 'komputer']	1
9	['point', 'private', 'mikrotik', 'ukm', 'radio']	['radio', 'point', 'to', 'point', 'line', 'of', 'sight', 'openvpn', 'metode', 'tun', 'metode', 'tap']	0,4
10	['ibu', 'hamil', 'asupan', 'gizi', 'genetika']	['algoritma', 'genetika', 'optimasi', 'gizi', 'ibu', 'hamil']	0,8
11	['pbb', 'jombang', 'pajak', 'framework', 'codeigniter']	['pbb', 'sistem', 'lapor', 'dan', 'olah', 'data', 'reuse-oriented', 'software', 'engineering']	0,2
12	['inference', 'metabolisme', 'angka', 'system', 'basal']	['algoritma', 'genetika', 'optimasi', 'fungsi', 'anggota', 'hitung', 'amb', 'genetic', 'algorithms', 'optimization', 'of', 'membership', 'function', 'calculation', 'of', 'amb']	0
13	['company', 'kembang', 'clothing', 'clove', 'supplier']	['pilih', 'pasok', 'ahp', 'sistem', 'dukung', 'putus']	0
14	['abadi', 'pasang', 'mode', 'gagal', 'baik']	['sistem', 'dukung', 'putus', 'fuzzy', 'fmea', 'rca']	0

15	['testing', 'tiga', 'prototype', 'putra', 'usability']	['sistem', 'informasi', 'customer', 'relationship', 'management', 'bisnis', 'mebel', 'marketing', 'automation', 'customer', 'sup']	0
16	['code', 'inventaris', 'qr', 'ptiik', 'informasi']	['inventaris', 'ptiik', 'qr', 'code', 'android']	0,8
17	['sedia', 'arima', 'pustaka', 'subjek', 'buku']	['data', 'warehouse', 'times', 'series', 'rolap', 'arima']	0,2
18	['minta', 'jumlah', 'koran', 'extreme', 'prediksi']	['koran', 'extreme', 'learning', 'machine', 'mean', 'square', 'error', 'mse']	0,4
19	['cegah', 'faktor', 'serviks', 'kanker', 'resiko']	['expert', 'system', 'kanker', 'serviks', 'certainty', 'factor', 'forward', 'chaining', '']	0,4
20	['transportasi', 'floyd-warshall', 'umum', 'alat', 'jalur']	['sistem', 'dukung', 'putus', 'transportasi', 'umum', 'kota', 'malang', 'jalur', 'pendek', 'floyd', 'warshall']	0,6
21	['promethee', 'psikologi', 'layan', 'umm', 'plp']	['pegawai', 'marketing', 'promethee', 'ranking']	0,2
22	['didik', 'uptd', 'cabang', 'dinas', 'sertifikasi']	['analytical', 'hierarchy', 'process', '', 'sistem', 'dukung', 'putus', '', 'uji', 'akurasi', '', 'uji', 'sensitivitas', '', 'sertifi']	0
23	['miskin', 'tangga', 'rumah', 'product', 'proses']	['bahan', 'bakar', 'minyak', 'rumah', 'tangga', 'miskin', 'weighted', 'product', 'java']	0,8
24	['balancing', 'least', 'connection', 'load', 'server']	['load', 'balancing', 'algoritma', 'jadwal', 'server', 'web', 'least', 'connection', 'software', 'defined', 'network']	1
25	['komposisi', 'derita', 'makan', 'mellitus', 'diabetes']	['algoritma', 'genetika', 'diabetes', 'mellitus', 'komposisi', 'makan', 'kolesterol']	0,8
26	['subtractive', 'bangkit', 'atur', '*cluster*ing', 'beasiswa']	['beasiswa', 'dikti', 'subtractive', '*cluster*ing', 'ekstraksi', 'atur', 'fuzzy', 'inference', 'system', 'model', 'sugeno', 'deteksi', 'stroke']	0,8
27	['customer', 'toko', 'relationship', 'buku', 'electronic']	['teknologi', 'customer', 'relationship', 'management', 'crm', 'prototyping', 'sistem', 'informasi', 'cutomer', 'otomatisasi', 'pasar', 'layan']	0,4
28	['path', 'first', 'shortest', 'open', 'routing']	['ospf', 'sdn', 'abilene', 'qos', 'voip', 'mesh', 'network', 'wireless', 'voice']	0
29	['swarm', 'optimization', 'particle', 'inferensifuzzy', 'jurus']	['particle', 'swarm', 'optimization', 'hepatitis', 'sistem', 'inferensi', 'fuzzy', 'mamdani', 'optimasi',	0,8

		'fungsi', 'anggota', 'tentu', 'jurus', 'siswa', 'sma', 'part']	
30	['evaluation', 'heuristic', 'usability', 'situs', 'evaluasi']	['heuristic', 'evaluation', 'persona', 'usability', 'uji', 'preferensi', 'user', 'preferences', 'testing']	0,6
31	['similarity', 'technique', 'ideal', 'preference', 'for']	['ahp', 'sehat', 'simplisia', 'sistem', 'dukung', 'putus', 'topsis', 'data', 'mining', 'klasterisasi', 'algoritma', 'k-means', 'improved', 'weighted', 'average']	0
32	['gula', 'sidoarjo', 'candi', 'pg', 'baru']	['gula', 'produksi', 'amal', 'extreme', 'learning', 'machine', 'rolap', 'lapor', 'exponential', 'smoothing', 'mape', 'data', 'warehouse']	0,2
33	['jadwal', 'semester', 'awas', 'awat', 'fakultas']	['algoritma', 'genetika', 'jadwal', 'awat', 'awas', 'uji', 'semester']	0,8
34	['series', 'time', 'tampung', 'hari', 'saham']	['average-based', 'fuzzy', 'time', 'series', 'logic', 'relationship', '', 'interval', 'efektif', 'algoritma', 'genetika', 'jaring', 'syaraf', 'tiru', 'backpropagation']	0,4
35	['tolerant', 'delay', 'konsultasi', 'dtn', 'tukar']	['delay', 'tolerant', 'network', 'telehealth', 'rural', 'ibr-dtn', 'raspberry', 'pi']	0,4
36	['quantization', 'lvq', 'vector', 'learning', 'gigi']	['klasifikasi', 'sakit', 'gigi', 'dan', 'mulut', 'learning', 'vector', 'quantization', 'mutu', 'susu', 'sapi', 'pecah', 'data', 'transformasi', 'association', 'rule', 'basis', 'asosiasi', 'classification', 'mining', 'jaring', 'syaraf', 'tiru', 'vect']	0,8
37	['k-means', 'kanker', 'protein', 'jenis', 'susun']	['kanker', 'payudara', 'metode', 'klustering', 'algoritma', 'na ve', 'bayes', 'k-means', 'knn', 'data', 'mining', 'bioinformatika', 'protein']	0,6
38	['ayam', 'ahp', 'broiler', 'isi', 'telur']	['ayam', 'ras', 'telur', 'ahp', 'fuzzy', 'topsis', 'akurasi', 'preferensi', 'layak', 'sistem', 'dukung', 'putus']	0,6
39	['kuliner', 'rute', 'kota', 'lokasi', 'angkut']	['aplikasi', 'tentu', 'rute', 'lokasi', 'kuliner', 'di', 'kota', 'malang', 'bas', 'gis', 'guna', 'metode', 'a', 'restoran', '', 'sistem', 'rekomendasi', 'weighted', 'product', 'optimasi', 'angkut', 'algoritma', 'genetika']	1
40	['diagnosa', 'anak', 'naive', 'pakar', 'kulit']	['pasti', 'sakit', 'kulit', 'sistem', 'pakar', 'akurasi validasi', 'na ve', 'bayes', 'demam', 'darah', 'naive', 'certainty', 'factor', 'ternak',	0,8

		'diagnosa', 'sapi', 'anjing', 'metode', 'bayesian', 'network']	
41	['modified', 'neighbor', 'tanam', 'diagnosa', 'k-nearest']	['sistem', 'pakar', 'modified', 'k-nearest', 'neighbor', 'mknn', 'sakit', 'tanam', 'cabai', 'merah', 'klasifikasi', 'kucing', 'dokumen', 'obat', 'dan', 'metode', 'nwknn']	0,8
42	['informatika', 'teknik', 'video', 'lan', 'profile']	['android', 'video', 'streaming', 'delay', 'jitter', 'packet', 'loss', 'throughput', 'profile', 'matching', 'seleksi', 'asisten', 'praktikum', 'sistem', 'dukung', 'putus']	0,4
43	['manajemen', 'program', 'gaji', 'unjung', 'taraperkasa']	['sistem', 'informasi', 'manajemen', 'pegawai', 'untuk', 'gaji', 'usaha', 'fungsional', 'uml', 'unjung', 'kenal', 'wajah', 'reuse', 'oriente']	0,6
44	['anp', 'analytic', 'guru', 'process', 'maospati']	['nilai', 'kerja', 'guru', 'analytic', 'network', 'process', 'peringkat', 'sistem', 'dukung', 'putus', 'bidang', 'studi', 'inap', 'analyic', 'urut', 'prioritas', 'analytical', 'kontraktor', 'proyek', 'kriteria', 'sensitivitas']	0,6
45	['smartphone', 'android', 'gedung', 'rancang', 'bangun']	['android', 'googlemaps', 'api', 'global', 'positioning', 'system', 'pandu', 'informasi', 'augmented', 'reality', '', 'gps', 'koordinat', 'webserver', 'sistem', 'wisata', 'malang', 'raya', 'mobile', 'jejaring', 'sosial', 'microbloging', 'autogeotagging', 'request', 'per', 'second', 'gadai', 'simulasi', 'e-learning', 'moodle', 'web', 'service', 'rest', 'moodle mobile', 'google', 'app', 'engine', 'cloud', 'sql', 'kml', 'social', 'networking']	0,2
46	['jantung', 'pjk', 'fk-nn', 'koroner', 'tingkat']	['improved', 'k-means', 'kelompok', 'data', 'tingkat', 'sehat', 'jantung', 'sakit', 'koroner', 'pjk', 'mining', 'logika', 'fuzzy', 'k-nearest', 'neighbor', 'fk-nn']	1
47	['suara', 'padu', 'bbp-ppa', 'beasiswa-ppa', 'tsukamoto']	['padu', 'suara', 'fuzzy', 'ahp', 'fuzzy-ahp', 'algoritma', 'genetika', 'fis', 'tsukamoto', 'beasiswa']	0,6
48	['support', 'jalan', 'rel', 'vector', 'svm']	['tsunami', 'klasifikasi', 'support', 'vector', 'machine', 'sakit', 'stroke', 'svm', 'simplified', 'sequential', 'minimal', 'optimizati', 'classification', 'optimization', 'rel', 'training', 'genre', 'musik', 'fuzzy']	0,8

49	['bahasa', 'indonesia', 'sentimen', 'twitter', 'neighbor']	['analisis', 'sentimen', 'klasifikasi', 'teks', 'neighbor-weighted', 'k-nearest', 'neighbor', 'sentiment', 'analysis', 'text', 'classification', 'mining', 'improved', 'knn', 'kategori', 'naive', 'bayes', 'twitter', 'stemming', 'bahasa', 'indonesia']	1
50	['hama', 'digital', 'deteksi', 'tanam', 'citra']	['apel', 'sakit', 'daun', 'otsu', 'support', 'vectore', 'machine', 'olah', 'citra', 'digital', 'principal', 'component', 'analysi', 'waterfall', 'sistem', 'pakar', 'bas', 'web', 'certainty', 'factor', 'cf', 'dempster-shafer', 'hama', 'tanam', 'mangga']	0,8
51	['weighting', 'simple', 'additive', 'fuzzy-simple', 'mandiri']	['sistem', 'dukung', 'putus', 'saw', 'pencak', 'silat', 'seleksi', 'spk', 'fmadm', 'fuzzy', 'simple', 'additive', 'weighting', 'fsaw', 'terima', 'pegawai', 'fuzzy-simple', 'addictive', 'f-saw', 'obat', 'awat', 'kulit', 'wajah']	0,8

Hasil *precision* tiap-tiap *cluster* menunjukan nilai *precision* tertinggi yaitu pada *cluster* 8, 24, 39, 46, dan 49 dengan nilai *precision* 1. *Cluster* tersebut artinya seluruh label yang diperoleh relevan dengan kata kunci yang ada. Sedangkan untuk *cluster* 12, 13, 14, 15, 22, 28 dan 31 memiliki nilai *precision* 0 yang artinya seluruh label yang diperoleh dari *cluster* tersebut tidak sesuai dengan kata kunci dokumen skripsi. Nilai *precision* keseluruhan *cluster* diperoleh dari rata-rata nilai *precision* dari seluruh *cluster* sehingga nilai *precision* keseluruhan yaitu 0,5153.

BAB 7 PENUTUP

7.1 Kesimpulan

Berdasarkan pengujian dan analisis yang dilakukan, dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Metode *Hierarchical Agglomerative Clustering* lebih sering menghasilkan *singleton* (*cluster* yang terdiri dari 1 dokumen) sehingga mempengaruhi ketepatan suatu *cluster* dalam mengelompokkan dokumen. Dari 3 parameter pemilihan jarak (*linkage*) dapat disimpulkan *average linkage* lebih baik dalam mengelompokkan dokumen dibandingkan dengan *single linkage* dan *complete linkage.*
2. Nilai *silhouette coefficient* tertinggi dari pengujian 100 dokumen skripsi diperoleh dari parameter *average linkage* pada titik potong 48 dengan jumlah *cluster* sebanyak 52 *cluster* dengan nilai *silhouette coefficient* yaitu 0,160428. Tingkat kekuatan yang dihasilkan dari pengujian dengan *Silhouette Coefficient* dapat dipengaruhi oleh beberapa hal yaitu dokumen yang digunakan, jumlah dokumen, jumlah *cluster* yang dihasilkan, jumlah *singleton* yang ada.
3. Label *cluster* yang dihasilkan dengan pembobotan *TF-IDF* membuat label menjadi unik karena dengan *TF-IDF term* yang dihasilkan sebagai label adalah *term* yang sering muncul pada *cluster* tersebut namun tidak banyak muncul di *cluster* lain. Jika label *cluster* dibandingkan dengan kata kunci dokumen skripsi maka rata-rata nilai *precision* yang dihasilkan dari titik potong 48 dengan jumlah *cluster* sebanyak 52 *cluster* yaitu 0,5153.

7.2 Saran

Dari kesimpulan yang diperoleh, ada beberapa saran yang dapat diberikan untuk pengembangan penelitian ini, diantaranya:

1. Menggunakan metode ekstraksi fitur lain yang bisa mendapatkan fitur yang lebih baik sehingga dapat meningkatkan ketepatan dari *cluster* dalam mengelompokkan dokumen.
2. Menggunakan parameter jarak (*linkage*) lain untuk penggabungan dokumen sehingga memungkinkan untuk mendapatkan nilai evaluasi yang lebih tinggi.
3. Menggunakan metode pelabelan lain untuk mendapatkan hasil label *cluster* yang lebih spesifik dan unik serta tepat mewakili dokumen pada *cluster.*
4. Menggunakan data berupa judul dan abstrak skripsi.

DAFTAR PUSTAKA

Feldman, R., & Sanger, J. (2007). *The Text Mining Handbook Advanced Approaches in Analyzing Unstructured Data.* Cambridge: Cambridge University Press.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining: Concepts and Techniques* (3rd ed.). Elsevier.

Handoyo, R., Mangkudjaja, R., & Nasution, S. M. (2014). Perbandingan Metode *Cluster*ing Menggunakan Metode Single Linkage dan K-Means pada Pengelompokan Dokumen. *JSM STMIK Mikroskil, 15*, 73-82.

Huda, M. (2011). Perkembangan Keilmuan di STAIN ponorogo. *Jurnal Dialogia, 9*.

Jaiswal, A., & Janwe, P. N. (2011). Hierarchical Document *Cluster*ing: A Review. *International Journal of Computer Applications*, 37-41.

Manning, C. D., Raghavan, P., & Schutze, H. (2009). *An Introduction to Information Retrieval.* Cambridge: Cambridge UP.

Nunes, B. P., Mera, A., Kawase, R., Fetahu, B., Casanova, M. A., Casanova, M. A., & Campos, G. H. (2014). A Topic Extraction Process for Online Forums. (pp. 541-543). Athens: IEEE 14th International Conference on Advanced Learning Technologies.

Popat, S. K., Deshmukh, P. B., & Metre, V. A. (2017). Hierarchical document *cluster*ing based on cosine similarity measure. (pp. 153-159). Aurangabad: 1st International Conference on Intelligent Systems and Information Management (ICISIM).

Pradnyana, G. A., & Sanjaya, N. A. (2012). Perancangan Dan Implementasi Automated Document Integration Dengan Menggunakan Algoritma Complete Linkage Agglomerative Hierarchical *Cluster*ing. *Jurnal Ilmu Komputer, 5*(2).

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of *cluster* analysis. *Journal of Computational and Applied Mathematics*, 53-65.

Saad, F. H., Mohamed, O. I., & Al-Qutaish, R. E. (2012). Comparison of Hierarchical Agglomerative Algorithms For *Cluster*ing Medical Documents. *International Journal of Software Engineering & Applications (IJSEA), 3*.

Sahu, L., & Mohan, M. B. (2014). An improved K-means algorithm using modified cosine distance measure for document *cluster*ing using Mahout with Hadoop. *2014 9th International Conference on Industrial and Information Systems (ICIIS)*.

Sandhya, N., Lalitha, Y. S., Govardhan, D. A., & Anuradha, D. K. (2008). Analysis of Similarity Measures for Text *Cluster*ing. *International Journal of Design Engineering (IJDE), 2*(4).

Umamaheswari, R., & Rajesh, K. (2014). Text *Cluster*ing Using Cosine Similarity and Matrix Factorization. *International Journal of Research in Computer and Communication Technology, 3*(10), 1343-1347.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Elsevier.